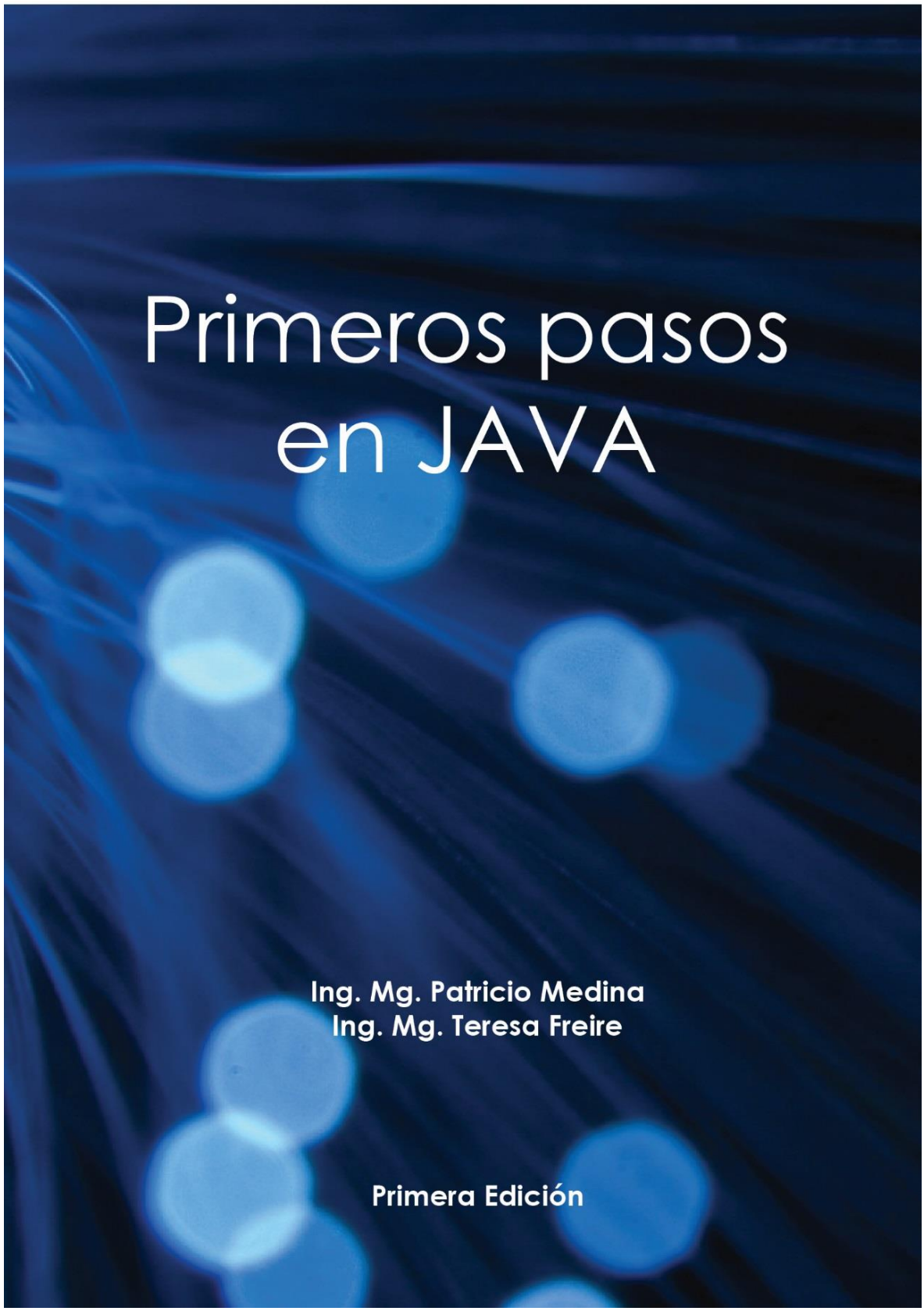

Primeros pasos
en JAVA
Ing. Mg. Patricio Medina
Ing. Mg. Teresa Freire
Primera Edición

PRIMEROS PASOS

EN JAVA

Ing. Mg. Patricio Medina

Profesor de la Universidad Técnica de Ambato
Profesor de la Pontificia Universidad Católica del Ecuador Sede Ambato
Asesor Informático

Ing. Mg. Teresa Freire
Profesor de la Pontificia Universidad Católica del Ecuador Sede Ambato

Primera Edición

Versión Digital

Autor: Ing. Mg. Patricio Medina Ch.

Capacitación realizada en: Cuba, Brasil y Ecuador

Contacto: ricardopmedina@uta.edu.ec, pmedina@pucesa.edu.ec, medina_patricio@yahoo.es

Autor: Ing. Mg. Teresa Freire A.

Contacto: tfreire@pucesa.edu.ec

ISBN- 978-9942-21-158-3

Primera Edición

Ambato – Ecuador

2015

DEDICATORIA

El que no hace un esfuerzo para ayudarse a sí mismo, no tiene derecho a solicitar ayuda a los demás. ***Demóstenes***

- A mi familia, mi más grande motivación y mi razón de existir.

Teresa

- A mi familia y estudiantes.

Patricio

Un agradecimiento especial a
María Fernanda Morales V. y
Santiago Monge L. por su valiosa ayuda.

INTRODUCCIÓN

La resolución de problemas mediante programas computacionales es un reto para quienes desean introducirse en el amplio mundo del desarrollo de aplicaciones.

Quienes se inician en el campo de la programación deben estar conscientes de que el obtener resultados óptimos con un programa informático requiere de la aplicación de un proceso metodológico adecuado que permita resolver un problema de forma lógica y plasmar la solución a través de un lenguaje de programación.

Dentro del presente texto se introduce al lector en el manejo del marco conceptual requerido para dar sus primeros pasos en el desarrollo de programas informáticos en el lenguaje de programación java, así también, se lo orienta en la aplicación de un proceso sistemático que inicia con el análisis del problema, el diseño de la solución, el desarrollo de la aplicación y las pruebas que validan los resultados obtenidos, temas que fortalecen el desarrollo del pensamiento lógico, requisito indispensable para un programador.

Además se presenta un conjunto amplio de ejercicios que van incrementando el nivel de dificultad y abarcan temas como el manejo de sentencias de entrada – salida, sentencias de control, manejo de arreglos, la programación orientada a objetos y el manejo de estructuras de datos, áreas indispensables en el desarrollo de programas en cualquier lenguaje de programación.

Estamos seguros de que la información presentada será de gran ayuda para los estudiantes de las áreas de programación y para quienes deseen introducirse en el desarrollo de aplicaciones, convirtiéndose en una herramienta de ayuda para dar sus primeros pasos en Java.

PRÓLOGO

La humanidad vive en un ciclo en el cual el ser humano no solo busca la respuesta a las múltiples preguntas que continuamente se formula, sino que también se encuentra en búsqueda de nuevas preguntas para contestar.

El trabajo que tienen ante ustedes es la gran obra de Patricio Medina y Teresa Freire, y es el resultado de años de intensas investigaciones en el campo académico y tecnológico, con respecto a la lógica de programación, para responder a la necesidad de un material de fácil acceso y manejo para la juventud que se inicia en sus estudios.

Es grato ofrecerles "PRIMEROS PASOS EN JAVA", con una selección de temas que puede servir de orientación y guía a los estudiantes que inician en el campo de la programación de computadores y para todas las personas interesadas en aprender un nuevo lenguaje de programación, y talleres de ejercicios de programación que le facilita el análisis, la codificación y el funcionamiento del ejercicio.

En estas páginas se ha acopiado las lecciones sistemáticas de un material realmente difícil y, el texto, a la par que sencillo, impone un fondo importante de conocimientos y actividades para desarrollar aplicaciones básicas en lógica de programación, con procesos intuitivos secuenciales y de fácil aplicación.

Se tiene la convicción plena de que sabrán apreciar este trabajo como un nuevo instrumento educativo; no lo desprecie. Recuerde que a través del lenguaje de programación, usted tiene la oportunidad de desarrollar su pensamiento crítico-científico y responder efectivamente a los diversos y complejos problemas de nuestra realidad nacional.

Ahora que tengo el privilegio de presentar el texto intitulado: "PRIMEROS PASOS EN JAVA", es oportunidad para felicitar el esfuerzo de los autores.

Dra. Teresa Hidalgo Lozada

DOCENTE UNIDAD EDUCATIVA "HISPANO AMÉRICA"

INDICE

INTRODUCCIÓN A LOS SISTEMAS COMPUTACIONALES

1. SISTEMA COMPUTACIONAL

Un proceso de computación consiste en la **transformación de datos en información**. Los datos constituyen los hechos ingresados al computador para su procesamiento mientras que la información es el resultado del procesamiento de dichos datos en una forma que es útil al usuario del sistema computacional que realiza el proceso de transformación también llamado procesamiento de datos.

El procesamiento de datos para generar información requiere de dos componentes fundamentales interactuando entre sí: ***el hardware y el software***. Dichos componentes configuran un sistema computacional.

El hardware de un sistema computacional son todos sus componentes electrónicos y electromecánicos, Así por ejemplo: el teclado, el mouse, el monitor, los discos duros, la memoria, el procesador, las unidades de cinta, las unidades de diskettes, son algunos ejemplos de componentes hardware.

En cambio, ***el software*** está constituido por los programas y la documentación asociada a éstos que especifican la forma en que los componentes de hardware son utilizados para realizar una cierta tarea tal como la generación impresa de los cheques de sueldos para los empleados de una empresa, el proveer el soporte necesario para poder accesar las páginas WWW de un curso de educación a distancia, o el escribir un informe. El software se lo puede categorizar en sistemas operativos, aplicaciones y en lenguajes de programación.

La interrelación entre los componentes hardware y software de un sistema computacional es tal, que el sistema sólo puede ser concebido como tal si ambos elementos están presentes simultáneamente. Los programas que integran el software son listas de instrucciones que describen procesos de computación específicos y la acción de construir dichas listas de instrucciones es lo que denominamos programación.

Un lenguaje de programación de alto nivel está orientado hacia el usuario del sistema computacional, simplificando la labor de programación y prueba. En cambio, un lenguaje de bajo nivel o lenguaje de máquina expresa directamente las capacidades y características del hardware, facilitando de ese modo su explotación, a costa de una mayor complejidad en el proceso de desarrollo de programas.

Un programa es una descripción estática de un proceso de computación. Para que dicho proceso pueda llevarse a cabo, las instrucciones que componen dicho programa, así como también los datos, deben ser almacenados al interior del sistema computacional, en lo que se denomina la memoria del sistema. Desde allí las instrucciones son tomadas una a una por otro componente del sistema computacional denominado procesador, quién es el encargado de su ejecución, es decir, de la concreción de las acciones especificadas a través de las instrucciones del programa.

Los componentes que pertenecen al área de procesamiento se sitúan sobre la placa madre (también denominada placa principal) de la computadora. Se usa el término placa madre debido a que todos los demás grupos de componentes y dispositivos periféricos son controlados a través de la misma.

Con la excepción de los puertos de entrada y salida de datos y el dispositivo de almacenamiento masivo, que de hecho es un periférico, la placa madre constituye la computadora en sí, por cuanto el procesamiento o el tratamiento de los datos tiene lugar siempre sobre la placa madre.

El ingreso de datos al sistema computacional es realizado por los usuarios del sistema vía dispositivos de entrada tales como teclados, mouse, pantallas sensibles al tacto, joysticks, analizadores de voz, etc. De modo similar, la información generada por el sistema computacional es transferida al usuario vía dispositivos de salida tales como monitores de vídeo, impresoras, sintetizadores de voz, etc.

Ciertos dispositivos tales como las unidades de discos magnéticos, los lectores/grabadores de CD-ROM, las unidades de diskette y cinta magnética, etc., pueden actuar como dispositivos de entrada y salida, siendo su propósito fundamental el almacenamiento masivo de datos. La utilidad de estos dispositivos radica en que liberan al usuario de la necesidad de ingresar todos los datos requeridos para realizar un proceso de computación cada vez que dicho proceso se realiza.

En un proceso de computación típico, sólo algunos de los datos son ingresados por el usuario, otros han sido previamente almacenados en medios de almacenamiento magnéticos, ópticos o de otra naturaleza, los que pueden ser leídos por dispositivos orientados al almacenamiento masivo de datos (lo cual también requiere de la realización de un proceso de computación).

No todos los datos son accesibles desde dispositivos de almacenamiento directamente conectados al computador que Ud. Está utilizando. En ocasiones, algunos de ellos residen en medios de almacenamiento accesibles por dispositivos de almacenamiento pertenecientes a otro sistema computacional. En esos casos, el acceso a dichos datos requiere de la interconexión entre ambos sistemas computacionales, configurando lo que se denomina una red de computadores.

2. RESOLUCIÓN DE PROBLEMAS MEDIANTE PROGRAMAS COMPUTACIONALES

2.1 SOLUCIÓN DE PROBLEMAS

Un computador es una "herramienta" de apoyo, lo que causa que por sí sola no sea capaz de solucionar un determinado problema.

Uno de los objetivos de la creación de los computadores fue el tratamiento masivo de información, lo que es posible gracias a que los computadores están en capacidad de recibir, almacenar y procesar información pero en base a información complementaria que son los llamados programas

Un programa entonces es un conjunto ordenado de instrucciones expresadas en un lenguaje de programación, que le indican a la máquina qué hacer, en qué orden y con qué datos.

ALGORITMIZACIÓN

Es el estudio sistemático de las técnicas fundamentales usadas para el diseño y análisis eficiente de algoritmos.

2.2 ALGORITMO

Es el proceso, método, técnica o conjunto finito de reglas que dan una sucesión de operaciones para resolver un problema específico, en otras palabras es un conjunto de pasos ordenados que nos van a permitir obtener la solución de un problema determinado.

Ejemplo: Para calcular el promedio de calificaciones de un número de estudiantes:
1. De dónde tomar los datos
2. Valores de las notas y número de estudiantes.
3. Se deben considerar notas ausentes.
4. Calcular el promedio
5. Emitir el resultado.

2.3 PROPIEDADES DEL ALGORITMO

- **FINITUD:** Un algoritmo debe terminar luego de un número finito de pasos.
- **DEFINIBILIDAD:** Cada paso del algoritmo debe especificarse de forma clara y precisa.
- **ENTRADA:** Un algoritmo puede tener cero o más entradas, es decir puede tener cantidades asignadas antes de iniciar el algoritmo y se denominan conjuntos especificados de objetos.

- **SALIDA:** Un algoritmo tiene una o más salidas, es decir cantidades que tienen relación con las entradas.
- **EFECTIVIDAD:** Las operaciones a realizar deben ser básicas para que se efectúen de modo exacto y en un tiempo dado con papel y lápiz.

2.4 PASOS PARA RESOLVER UN PROBLEMA

1. IDENTIFICAR UN PROBLEMA:

Significa el planteamiento del problema, lo que se conoce como el enunciado del mismo.

2. ESTUDIO DEL PROBLEMA Y ANÁLISIS

Se requiere conocer el objetivo real del programa, y se determina lo que debe hacer y el resultado o solución deseada.

Para definir bien un problema, es necesario responder a las preguntas:

- ¿Qué entradas se requieren? (tipo y cantidad)
- ¿Cuál es la salida deseada? (tipo y cantidad)
- ¿Qué método produce la salida deseada?

3. DISEÑO DE LA SOLUCIÓN

Es el detalle de las operaciones lógicas que el computador deberá realizar para obtener el resultado deseado. Es independiente del computador y del lenguaje a utilizar.

4. PRUEBA DE LA SOLUCIÓN

Comprobar que el diseño cumpla las especificaciones detalladas, creando información de entrada con la que se comprueban los pasos del diseño y verificar que los resultados sean los correctos.

5. CODIFICACIÓN

Es la traducción del diseño al lenguaje de programación, utilizando las reglas propias de cada lenguaje (C, Visual Basic, Delphi, etc.)

6. PUESTA A PUNTO

Es similar a la prueba de la solución, sólo que ésta se efectúa en el computador.

2.5 TÉCNICAS DE DISEÑO

Cada programador puede utilizar la metodología que mejor se adapte a sus necesidades, dependerá principalmente de su experiencia. Para el diseño de la solución las principales metodologías son:

1. ***Sistema Narrativo, ALGORITMO***, que es la explicación por medio de frases, de los pasos, operaciones, condiciones, etc., que se deben seguir para obtener la solución.

2. ***Sistema Gráfico, DIAGRAMA***, es la secuencia representada por medio de símbolos de las operaciones, condiciones, etc., que se deben seguir para obtener la solución.

   El uso de los diagramas se justifica por varias razones:

   1. Es independiente del lenguaje de programación (no hace falta conocerlo)
   2. Representa una clara descripción gráfica de la solución.
   3. Permite representar todas las estructuras imaginables, por complejas que sean.
   4. Sirven como documentación del problema.

3. TIPOS DE DATOS

Para ejecutar un programa necesitamos de datos, los que pueden ser de distinta naturaleza, así: Valores numéricos, Nombres, Direcciones, Notas de ventas, que dependen de la naturaleza del problema.

Los datos se clasifican en tres clases:

1. NUMÉRICOS

- **Enteros:** Números completos, ejemplo: número de notas, número de estudiantes.
- **Reales:** Números con punto decimal, ejemplo: notas de estudiantes, precios de productos.

2. CADENAS DE CARACTERES

Las cadenas de caracteres se conocen con el nombre de datos “string”. Aquí pueden considerarse las letras del alfabeto, cadenas de números (0..9) y caracteres especiales (@, #, %, $, &,*). Ejemplo: Direcciones, Nombres, números de cédula, fechas.

Los datos string para diferenciarlos de los otros datos, se representan entre comillas (“ ”), ejemplo: “Informática Básica”.

3. LÓGICOS

Estos pueden tomar dos valores posibles: Verdadero o Falso.

4. VARIABLES

Una variable se forma con la combinación de caracteres alfabéticos, numéricos o especiales. Por ejemplo, una variable puede ser: A, A1, NOTA, NOTA1, etc.

La característica de una variable es que va a ser utilizada en nuestro programa y va a tener un valor determinado (siempre tiene un valor), dicho valor puede ser numérico, una cadena de caracteres o un valor lógico.

4.1 REGLAS PARA DEFINIR VARIABLES

Cada lenguaje de programación tiene sus propias reglas de definición de variables, pero las reglas generales que aplican a la mayoría de los lenguajes son las siguientes:

1. Las variables deben iniciar con una letra, no con un número o con un carácter especial. Ejemplo:

VALOR1 →Correcto.
1VALOR →Incorrecto.

2. Un identificador de variable no debe tener espacios en blanco, deben escribirse palabras juntas con un carácter especial que las separe. Ejemplo:

VALOR_1 →Correcto.
VALOR 1 →Incorrecto.

3. Se debe tener control sobre la longitud de la variable, tomando en cuenta el lenguaje que se esté utilizando.

Una variable puede tomar valores de dos maneras:
Realizando una lectura: Leemos los valores iniciales de las variables conocidas y con estos datos calculamos la variable desconocida (es decir la que nos hace falta).

Realizando Operaciones: Calculando los valores de las variables dentro del programa

Ejemplo: Para calcular la hipotenusa de un triángulo rectángulo, la fórmula es: $a^2=b^2+c^2$ Necesitamos que b y c sean datos, estos valores deberán ser introducidos mediante una lectura de datos., mientras que el valor de a se obtendrá con una operación en la que intervienen b y c.

4.2 OPERACIÓN DE ASIGNACIÓN

Para representar una operación de asignación lo vamos a hacer con una flecha que apunte hacia la izquierda:

A ← 3 se interpreta como: "Asigne a A el valor de 3"

La asignación es una operación destructiva, que quiere decir que una variable puede tener cualquier cantidad de valores en un momento determinado.

La asignación de valores se puede realizar de varias maneras:

- Mediante una constante: A ← 0 X ← 1
- Mediante una expresión aritmética: T ← 200+150 T ← (A+5)/B

4.3 OPERADORES

Los operadores básicos que en forma general se aplican a la mayoría de los lenguajes de programación son:

- Suma +
- Resta -
- Multiplicación *
- División /

4.4 REGLAS DE PREFERENCIA

1. Primero Se realiza la exponenciación, luego la multiplicación y división con el mismo nivel de prioridad y por último la suma y la resta también con el mismo nivel de prioridad
   Ejemplo: A+B*C = A+(B*C)
2. Se pueden usar únicamente paréntesis para definir o especificar la precedencia o prioridad de las operaciones y se aplican las mismas reglas que en álgebra.
3. Los paréntesis internos se ejecutan primero.
4. En la computadora todas las operaciones se representan en una sola línea en forma horizontal

5. INSTRUCCIONES

Una instrucción es una unidad básica, elemental de un programa, consta de dos partes:

1. El código de operación (parte fija), que indica la acción a ejecutar.
2. El operando (parte variable), indica los datos que intervienen.

5.1 TIPOS DE INSTRUCCIONES

Se clasifican en:

- **INSTRUCCIONES DE ENTRADA / SALIDA**: son las que establecen la comunicación entre el CPU y el exterior, ejemplos: Leer, Grabar, Imprimir, Visualizar, Aceptar.

- **INSTRUCCIONES ARITMÉTICAS**: Las que permiten operar campos numéricos, ejemplos: Sumar, restar, multiplicar, dividir.
- **INSTRUCCIONES DE ARCHIVO**: Permiten preparar los archivos que se requieren, ejemplos: Abrir, Cerrar.
- **INSTRUCCIONES DE BIFURCACIÓN**: Permiten alterar el flujo normal de un programa, ejemplos: Si.....Entonces.....Sino.....Entonces
- **INSTRUCCIONES DE DECLARACIÓN**: Permiten reservar áreas de memoria para definir constantes, campos intermedios, etc.

6. FUNCIONES

Una Función se considera a un conjunto de instrucciones que permiten realizar un cálculo determinado, cabe señalar que una función puede variar de un lenguaje a otro.

Una función se representa como: NombreFuncion(argumentos), en donde los argumentos pueden ser constantes, variables, expresiones matemáticas, y generalmente se colocan entre paréntesis. Ejemplos:

- Raíz cuadrada SQRT(Num)
- Coseno COS(angulo)

7. EJEMPLOS DE ALGORITMOS

1. **Ejercicio: Calcular la suma de dos números.**
   a. Análisis
      ¿Qué entradas se requieren? numero1 numero2
      ¿Cuál es la salida deseada? suma
      ¿Cómo obtener lo que se busca? suma=numero1+numero2

   b. Diseño de la solución: técnica descriptiva – Algoritmo.

      1. Inicio
      2. titulo
      3. LEER numero1, numero2
      4. CALCULAR suma=numero1+numero2
      5. VISUALIZAR numero1, numero2, suma
      6. Fin

2. Calcular la hipotenusa de un triángulo rectángulo.

Datos:
a, b, Número de triángulos.

Variables:

a, b o Valores de los catetos del triángulo.
NTRIANG o Número de triángulos a
resolver. c o hipotenusa.

Proceso:

1. Inicio
2. Leer
NTRIANG 3.
Leer a, b
4. Calculo de la hipotenusa: c = (an2 + bn2)n(1/2)
5. Salida: c
6. ¿El proceso se realizó NTRINAG veces?
   Si → Fin
   No→ Ir a 2
7. Fin

LENGUAJE JAVA

1. INTRODUCCIÓN

Según su sitio oficial, Java es un lenguaje de programación y una plataforma informática que apareció en 1995 respaldada por la empresa Sun MicroSystems. Como lenguaje de programación, es rápido, seguro, fiable y tiene compatibilidad con equipos que van desde portátiles hasta grandes centros de datos.

Se utiliza para desarrollar aplicaciones de todo tipo adaptables a consolas de juegos, equipos de cómputo tradicionales, súper computadoras, inclusive teléfonos móviles y aplicaciones para Internet. (Java, 2015).

La descarga de Java es gratuita y se accede a través del llamado JRE (Java Runtime Enviroment) que está compuesto de la máquina Virtual de Java (JVM Java Virtual Machine), un conjunto de bibliotecas y otros componentes necesarios para que una aplicación escrita en lenguaje Java pueda ser ejecutada.

La JVM permite la ejecución del código de java, las bibliotecas en cambio son las que conforman la interfaz de programación del lenguaje (API Application Programming Interface) y las dos deben ser consistentes, es por esta razón que se descargan en conjunto.

Si lo que se desea es ejecutar las aplicaciones desarrolladas en lenguaje Java solo será necesario el JRE, mientras que para desarrollar nuevas aplicaciones en dicho lenguaje es necesario un entorno de desarrollo, denominado JDK (Java Development Kit), que además del JRE (mínimo imprescindible) incluye, entre otros, un compilador que permite compilar y ejecutar programas en lenguaje Java.

El JDK entonces incorpora:

- El lenguaje de programación
- Las bibliotecas estándar
- El compilador de Java
- El generador de documentación
- El depurador de programas
- El entorno de ejecución (JRE) que está compuesto por la JVM

2. ENTORNOS DE DESARROLLO

Existe un sinnúmero de entornos de desarrollo para Java de distintas empresas, en su mayoría de libre distribución, todos ellos orientados a facilitar las actividades básicas sobre el desarrollo de aplicaciones en Java, es decir la edición del programa fuente, la compilación y ejecución del mismo.

Así se pueden mencionar:

a. **NetBeans**: entorno de libre distribución patrocinado por Sun MicroSystems.
b. **Eclipse**: entorno genérico de desarrollo patrocinado por IBM.
c. **BlueJ**: entorno desarrollado en la Universidad de Kent.
d. **JCreator LE**: versión de uso gratuito

Todos estos entornos facilitan el desarrollo de programas siguiendo un proceso sencillo y secuencial y utilizando las utilidades que ofrece el lenguaje, así se puede mencionar:

- La **Codificación** del programa fuente: logrado a través del editor del lenguaje de programación, permite crear el archivo de código fuente que en Java se identifica por un **nombre.java**. Cabe indicar que el nombre debe corresponder con el nombre de la clase.
- La **Compilación**, que permite comprobar o verificar la sintaxis del código fuente y transformarlo en el archivo objeto (bytecode). Este proceso le advertirá al programador de posibles errores detectados, los mismos que deberán ser corregidos antes de obtener el archivo con el mismo nombre pero con la extensión **.class**
- La **Ejecución**, que convierte el archivo objeto en una versión que permite comprobar la ejecución misma del programa.

3. ESTRUCTURA DE UN PROGRAMA EN JAVA

```
/**
 * @(#)prueba.java
 *
 * prueba application
 *
 * @author
 * @version 1.00 2015/5/5
 */

public class prueba
{
   public static void main(String[] args)
  {
        // cuerpo de código
        System.out.println("Hola a todos!");
   }
}
```

En este fragmento de código se pueden observar los elementos básicos de un programa en java. Java emplea siempre la Programación Orientada a Objetos por lo que todo el código se
incluye dentro de las llamadas clases.

CLASE

Una clase es un modelo o prototipo de información que permite definir objetos con similares características. Está compuesta de atributos y métodos y se declaran utilizando la forma básica siguiente:

[modificador de acceso] **class** nombre_clase
{
    Cuerpo de la clase
}

Los modificadores de acceso pueden ser public (una clase que puede ser usada en cualquier programa sin restricción), final (clase que no puede ser modificada ni heredada), abstract (clase que no puede ser instanciada).

En el ejemplo anterior, prueba es el nombre de la clase principal y del archivo que contiene el código fuente. Todos los programas o aplicaciones escritas en Java tienen un método main o principal que, a su vez, contiene un conjunto de sentencias. En Java los bloques de sentencias se indican entre llaves { }.

En el caso anterior, la sentencia usada es un método predefinido en Java: println () que permite visualizar texto en pantalla.

COMENTARIOS

En Java los comentarios pueden colocarse de dos maneras:

- Comentarios en una sola línea utilizando el //
- Comentarios en varias líneas utilizando el /* y cerrándolo con */

RESUMEN

Los elementos básicos de un programa en Java son:

a. Comentarios
b. Definición de la clase
c. Definición del método main
d. Sentencias

IDENTIFICADORES

Los identificadores son nombres que se les asignan a variables, métodos, clases en el código fuente de un programa. Todo nuevo identificador que se emplee en un programa Java debe definirse previamente a su utilización. Las normas para la construcción de un identificador empleando el lenguaje de programación Java son las siguientes:

- Deben utilizarse nombres significativos para que cuando se revise el código fuente se pueda recordar qué representa el identificador.
- Los nombres de las variables y los métodos deben comenzar con minúscula, si es un nombre compuesto cada palabra empieza con una letra mayúscula.
- Los nombres de las clases deben iniciar con mayúscula
- Los nombres de constantes deben escribirse en mayúsculas.
- Un identificador puede comenzar con una letra o con un carácter de subrayado
- No pueden ser nombres o palabras reservadas del lenguaje
- No hay límite de caracteres
- No pueden emplearse dos identificadores con el mismo nombre en el mismo bloque de código

TIPOS DE DATOS

Java tiene un conjunto de datos primitivos que permiten trabajar con información de distintos orígenes:

- Números enteros: almacena números enteros positivos y negativos
- Números reales: guarda números decimales
- Caracteres: guarda caracteres alfanuméricos
- Booleanos: guarda valores lógicos (true / false)

TIPO DE DATO	DECSRIPCIÓN
byte	Entero con signo
short	Entero con signo
int	Entero con signo
long	Entero con signo
float	Real con 6 dígitos de precisión
doublé	Real con 10 dígitos de precisión
char	Caracteres alfanuméricos
boolean	Verdadero o falso

VARIABLES

Permiten guardar valores de distintos tipos para ser trabajados dentro de un programa. Toda variable debe ser declarada antes de ser utilizada, para ello se utiliza el tipo y el identificador.

Los tipos de datos pueden ser elegidos de los de la lista anterior, el identificador debe responder a las normas antes mencionadas y finaliza la sentencia con punto y coma.

tipo identificador;

int numero;

CONSTANTES

La definición de una constante es similar a la de una variable anteponiendo el modificador final:

final tipo identificador = valor;

final double PI=3.14159;

Su valor no podrá cambiar de ninguna forma dentro del programa.

OPERADORES

Dentro del lenguaje Java se manejan distintos tipos de operadores que son el complemento para el desarrollo de las operaciones y para la estructuración de las instrucciones de control que más adelante serán explicadas.
Se tienen distintos tipos de operadores:

- Aritméticos
- Relacionales
- Lógicos

ARITMÉTICOS

Permiten estructurar las operaciones aritméticas básicas y pueden ser:

OPERADOR	USO
+	Operador suma
-	Operador resta
*	Operador producto
/	Operador división
%	Operador residuo de la división
++	Operador unario incremento
--	Operador unario decremento

RELACIONALES

Permiten estructurar expresiones de comparación, es decir, condiciones simples.

OPERADOR	USO
>	Mayor que
<	Menor que
>=	Mayor o igual

<=	Menor o igual
==	Igualdad
¡=	Diferencia

LÓGICOS

Sirven para estructurar expresiones lógicas, las mismas que se forman con dos o más condiciones simples.

OPERADOR	USO
&&	And
||	Or
¡	Not

EXPRESIONES

Se estructuran combinando variables de los distintos tipos de datos con los operadores. Siempre devuelven un valor resultante en función de la operación implementada.

ARITMÉTICAS

Por lo general son operaciones en las cuales intervienen dos o más números en conjunto con los operadores aritméticos. Siempre retornan como resultado otro número.

```
int x=10;
int y=2;
System.out.println("El residuo de la división: "+(x%y));
```

Se va a imprimir en pantalla el número cero.

DE RELACIÓN

Son aquellas que permiten comparar valores dando como resultado un valor booleano (true o false).

Tomando como base las variables del ejemplo anterior:

```
x>y
```

La respuesta sería true.

LÓGICAS

Se estructuran utilizando dos o más expresiones de relación acompañada de un operador lógico.

```
(x>y) && (x<=0)
```

Para obtener el resultado de la expresión es necesario revisar la información referente a las tablas de verdad, en base a lo cual el resultado de la expresión del ejemplo sería false.

ENTRADA DE DATOS DE VARIOS TIPOS

Existen varias formas en java para realizer el ingreso de datos por teclado, a continuación se detalla la forma más sencilla para los distintos tipos de datos primitivos:

```
import java.util.Scanner;
import java.io.IOException;

public class lecTiposDatos
{

   public static void main(String[] args) throws IOException
   {

   Scanner leer=new Scanner(System.in);


   //leer string
   String s;
   System.out.print("string: ");
   s=leer.next();
   System.out.println("string:"+s);

   //leer entero
   int i;
   System.out.print("numero entero: ");
   i=leer.nextInt();
   System.out.println("entero: "+i);

   //leer caracter
   char c;
   System.out.print("caracter: ");
   c=(char)System.in.read();
   System.out.println("caracter: "+c);

   //leer double
   double d;
   System.out.print("numero double: ");
   d=leer.nextDouble();
   System.out.println("double: "+d);

   //leer flotante
```

```
   float  f;
   System.out.print("numero flotante: ");
   f=leer.nextFloat();
   System.out.println("flotante: "+f);

   //leer booleano
   boolean  b;
   System.out.print("booleano: ");
   b=leer.nextBoolean();
   System.out.println("booleano: "+b);

   }
}
```

ESTRUCTURAS DE CONTROL

Java cuenta con dos tipos de estructuras de control: las estructuras de selección y las estructuras de repetición.

ESTRUCTURAS DE SELECCIÓN

Sirven para evaluar condiciones simples (implementadas con expresiones de relación), ó compuestas (implementadas con expresiones lógicas).

Las condiciones evalúan una expresión lógica o artitmético – lógica y retornan un resultado booleano.

Java implementa las estructuras de selección a partir de la sentencia if, la misma que trabaja de distintas maneras:

- **Sentencia if**
  Se estructura de la siguiente manera:

```
if(condición)
{
        sentencia1;
        sentencia2;
}
```

  Evalúa si la condición es verdadera en cuyo caso ejecuta las sentencias que se encuentran dentro del bloque de programa entre las {}. En el caso de que solo se tenga una única sentencia a ejecutar podrían omitirse las llaves.

- **Sentencia if-else**

  Se estructura de la siguiente manera:

```
if(condición)
{
```

```
        sentencia1;
        sentencia2;
}
else
{
        Sentencia3;
        Sentencia4;
}
```

Evalúa si la condición es verdadera o falsa, en cuyo caso ejecuta las sentencias que se encuentran dentro de los bloques de programa entre las {}.

- **Sentencia switch()**

Es una sentencia condicional compuesta que parte de la evaluación de una variable que puede tomar un valor y a partir de él ejecutar distintas instrucciones.

La sintaxis es la siguiente:

```
switch (variable)
{
    case valor1:
            instrucciones;
            break;
    case valor2:
            instrucciones;
            break;
    .
    .
    .
    default:
            sentencias;
            break;
 }
```

Dentro de los elementos que incluye se encuentran:

- ✓ La variable a evaluar que puede ser únicamente de un tipo de dato que almacene o valores enteros o tipo caracter.
- ✓ La palabra reservada case seguida de los valores constantes, permite comparar la variable con cada uno de dichos valores.
- ✓ La palabra reservada break permite que una vez que la variable dio como resultado que si es igual a una de las constantes, se ejecuten las sentencias correspondientes pero se rompa la ejecución del switch, ya que de los contrario java seguiría comparando la variables con las demás alternativas existentes.

- ✓ El default es opcional y permite ejecutar un conjunto de sentencias en el caso de que ninguno de los valores constantes especificados coincidan con la variable.

ESTRUCTURAS DE REPETICIÓN

Permiten ejecutar un conjunto de sentencias varias veces a partir del cumplimiento de ciertas condiciones de repetición.

En java se trabaja con las siguientes:

- **Sentencia while**

  Repite un conjunto de sentencias a partir de una condición evaluada al inicio de la instrucción.

  En vista de que la condición se verifica al inicio del ciclo, la sentencia podría ejecutarse 0 o más veces si es que de entrada la instrucción devuelve false.

  La estructura es la siguiente:

```
while (condición)
{
        Sentencia1;
        Sentencia 2;
}
```

  Es importante anotar que dentro del ciclo debe haber alguna instrucción que haga que cambie el comportamiento de la condición de lo contrario el ciclo se haría infinito.

- **Sentencia for**

  Este ciclo repetitivo se utiliza cuando se sabe de antemano el número de veces que se deberá repetir el ciclo, se le conoce como el ciclo automático, pues se repite en función de una variable que cambia directamente en la ejecución del ciclo.

  La estructura es la siguiente:

```
for(inicialización del contador; condición de repetición; incremento)
{
        Sentencia;
        Sentencia;
}
```

  La inicialización del contador corresponde a la instrucción en la cual la variable que va a intervenir en el ciclo se define y se inicializa con un valor. Esta variable puede ser de tipo numérica o de tipo carácter.

La condición de repetición permite evaluar que la variable contadora cumpla con la condición, permitiendo que siga ejecutándose el ciclo.

El incremento hace referencia al cambio de valor de la variable contadora.

El ciclo for puede ejecutarse 0 o más veces debido a que la condición de repetición se verifica desde la primera iteración del ciclo y es el incremento el que determina el cambio de la variables y el número de repeticiones que tendrá.

- **Sentencia do-while**

Esta sentencia es similar al while pero la condición de repetición se verifica al final.

Debido a ello, el ciclo podría ejecutarse al menos una vez, ya que independientemente de la condición las sentencias incluidas dentro del ciclo se ejecutarán la primera vez que se ingresa al mismo.

La estructura es la siguiente:

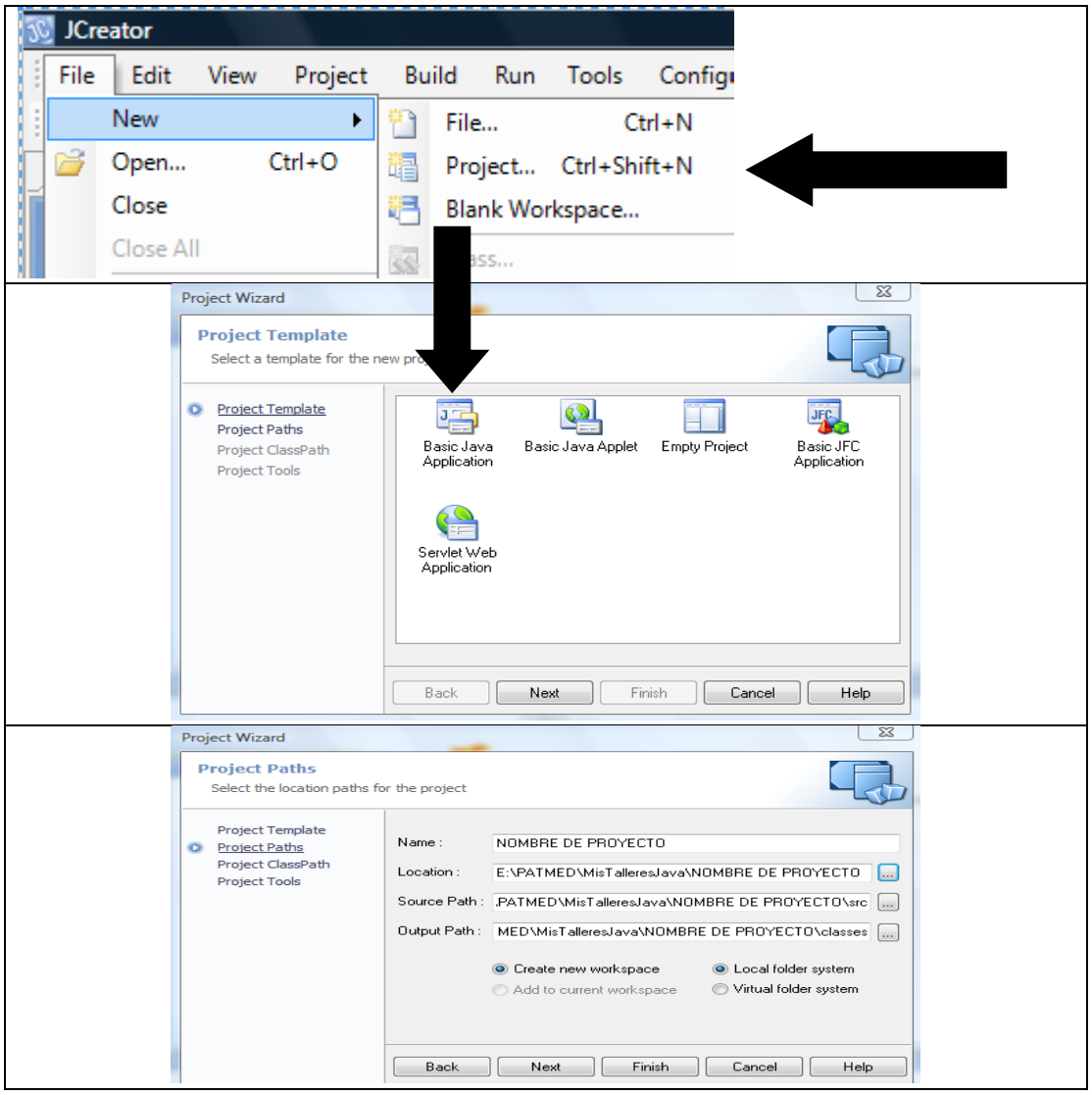
```
do
{
        Sentencia1;
        Sentencia 2;
}while(condición);
```

Tal como con el while, se debe procurar que la variable que interviene en la condición cambie dentro del ciclo de lo contrario se convertirá en un ciclo infinito.

EJEMPLOS DE CODIFICACIONES

TEMA 0: CREANDO UN NUEVO PROYECTO

1. Ingresar al programa Jcreator
2. Abrir un nuevo Proyecto
3. Seleccione Basic Java Application
4. Configure el **Nombre** (como se va llamar su proyecto) y la **Localización** (donde lo va a guardar)



5. EXPLICACIÓN

Este ejercicio, deberá hacerlo cada vez que empieza un nuevo proyecto de programación propuesto en este texto o alguno que desee realizar.

TEMA 1: INVERTIR UN NÚMERO DE DOS CIFRAS

1. Abrir un nuevo Proyecto
2. Código de la Clase

```
import java.util.Scanner;
public class Invertir2Num {

   public static void main(String[] args) {

        //Objeto para lectura de datos
        Scanner leer = new Scanner(System.in);

        int num, aux;
        int  dec, uni;
        System.out.print ("Ingrese un numero: ");
        num=leer.nextInt();
        System.out.println("\nUsted a ingresado el numero: " + num);
        dec = num / 10;
        uni = num % 10;
        aux = (uni * 10) + dec;
        System.out.println("\nEl numero invertido es: " + aux);

  }
}
```

3. **Grabar y ejecutar**

FUNCIONAMIENTO

El programa invierte un número entero de dos cifras, por ejemplo: si se ingresa el numero 21 el resultado será el número 12.

TEMA 2: INVERTIR UN NÚMERO DE TRES CIFRAS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Invertir3Cifras {

   public static void main(String[] args) {
        //Objeto para lectura de datos
                Scanner leer = new Scanner(System.in);

                int num, aux;
                int dec, uni, cen;
                System.out.print ("Ingrese un numero de tres cifras: ");
                num=leer.nextInt();
                System.out.println("El número de tres cifras: " + num);
                cen = num / 100;
                num = num % 100;
                dec = num / 10;
                uni = num % 10;
                aux = (uni * 100) + (dec * 10) + cen;
                System.out.println("El número invertido es: " + aux);
  }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa invierte un número entero de tres cifras, por ejemplo: si se ingresa el numero 381 el resultado será el número 183.

TEMA 3: PRACTICA DE OPERACIONES BÁSICAS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class OperacionesBasicas {

   public static void main(String[] args) {
        //Objeto para lectura de datos
                Scanner leer = new Scanner(System.in);

                float resul,num1,num2;
                System.out.print ("Ingrese un numero: ");
                num1=leer.nextFloat();
                System.out.print ("Ingrese el segundo número: ");
                num2=leer.nextFloat();
                resul = num1+num2;
                System.out.println("La Suma  es: "+resul);

                resul = num1-num2;
                System.out.println("La Resta es: "+  num1 + " - " + num2 + " = " + resul);

                resul = num1*num2;
                System.out.println("La Multiplicación es: "+ resul);

                resul =num1/num2;
                System.out.println("La División es: "+ resul);

                resul = num1 % num2;
                System.out.println("Residuo es: "+ resul);

                System.out.println("Gracias");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza las operaciones básicas como suma, resta, multiplicación, división y el residuo de dos números. Por ejemplo: se ingresa el numero 20 y 4 la suma es: 24, la resta : 16, la multiplicación: 80, la división: 5, y el residuo: 0.

TEMA 4: COMPRA EN RESTAURANTE

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Restaurant {

   public static void main(String[] args) {
        //Objeto para lectura de datos
                Scanner leer = new Scanner(System.in);

                int canB, canH, canP;
                double  aPagar;
                float precioB = 0.80F;
                float  precioH = 2.0F;
                float precioP = 1.25F;
                System.out.print ("Ingrese la cantidad de Hamburguesas: " );
                canH=leer.nextInt();
                System.out.print ("Ingrese la cantidad de Papas: ");
                canP=leer.nextInt();
                System.out.print ("Ingrese la cantidad de Bebidas: ");
                canB=leer.nextInt();
                System.out.println("Cantidad de Hamburguesas   :"+canH);
                System.out.println("Cantidad de Papas          :"+ canP);
                System.out.println("Cantidad de Bebidas        :"+canB);
                System.out.println();
                aPagar = (canH * precioH) + (canP * precioP) + (canB * precioB);
                System.out.println("Valor a Pagar: " + aPagar);

   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza el calculo de la cuenta total a pagar de la compra de hamburguesas, papas y bebidas en un restaurante, se ingresa por teclado la cantidad de cada compra.

TEMA 5: FUNCIONES BÁSICAS LIBRERÍA MATH

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class LibreriaMath {

   public static void main(String[] args) {
        //Objeto para lectura de datos
                Scanner leer = new Scanner(System.in);

                int num;
                long  resul;

                System.out.print("Ingrese un número  :");
                num=leer.nextInt();
                resul = Math.abs(num);
                System.out.println("VALOR ABSOLUTO : " + resul);
                System.out.println("POTENCIA       : " + Math.pow(resul, 3));
                System.out.println("RAIZ CUADRADA  : " + Math.sqrt(resul));
                System.out.println("SENO           : " + Math.sin(resul * Math.PI / 180));
                System.out.println("COSENO         : " + Math.cos(resul * Math.PI / 180));
                System.out.println("NÚMERO MÁXIMO  : " + Math.max(resul, 50));
                System.out.println("NÚMERO MÍNIMO  : " + Math.min(resul, 50));
                System.out.println("PARTE ENTERA   : " + Math.floor(18.78));
                System.out.println("REDONDEO       : " + Math.round(18.78));

  }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa nos enseña los diferentes metodos que se pueden usar con la librería Math, entre ellas estan valor absoluto, potencia, raiz cuadrada, seno, coseno, número máximo, número mínimo, parte entera y redondeo.

TEMA 6: OPERADORES PRE, POST INCREMENTO Y DECREMENTO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Incremento {

   public static void main(String[] args) {

        int i = 1;
                System.out.println("i : " + i);
                // Pre-incremento, primero incrementa y luego imprime por consola
                System.out.println("++i : " + ++i);

                // Post-incremento, primero imprime “2” por consola y luego incrementa i.
                System.out.println("i++ : " + i++);

                //i por lo tanto vale 3
                System.out.println("i : " + i);
                System.out.println();

                // Pre-decremento, primero Decrementa i y luego lo imprime por consola

                System.out.println("--i : " + --i);
                // Post-decremento, primero imprime i por consola y luego de decrementa.
                System.out.println("i-- : " + i--);
                // Ahora i vale 1
                System.out.println("i : " + i);

                System.out.println("Gracias");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa nos enseña los diferentes tipos de incrementos y decrementos, y como funcionan cada uno; el pre-incremento y pre-decremento primero realiza la operación respectiva y luego imprime, en cambio el post-incremento y post-decremento primero imprime y luego realiza la operación respectiva.

TEMA 7 CONVERSIÓN DE DATOS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Conversion {

   public static void main(String[] args) {

                int n1=Character.digit('7',10);
        int n2=1;

        Character letra=new Character('z');
double n3=150.56;

        String cad1="Numero: ";
                        String cad2=letra.toString();
        String cad=String.valueOf(n3);


        System.out.println(cad1+cad+cad2);
        System.out.println(n1+n2);

char nletra=Character.forDigit(n2,10);

System.out.print(n1+" "+nletra);

                        // REVISAR VARIOS CASOS EN EL FOLLETO
  }
}
```

3. **GRABAR Y EJECUTAR**

TEMA 8 MAYOR DE DOS NÚMEROS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Mayor2Num {

   public static void main(String[] args) {
        //Objeto para lectura de datos
                Scanner leer = new Scanner(System.in);

        int num1, num2;
                System.out.print("Ingrese el número 1  : ");
                num1=leer.nextInt();
                System.out.print("Ingrese el número 2  : ");
                num2=leer.nextInt();
                if (num1 > num2)
                         System.out.println(num1 +" es mayor que " + num2);
                else
                {
                        if (num1 == num2)
                                System.out.println(num1+ " es igual a "+num2);
                        else
                                System.out.println(num1+" es menor que "+num2);
                }
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa encuentra el mayor número de entre 2 números ingresados por teclado, por ejemplo si se ingresa el 15 y 28, el número mayor sería 28.

TEMA 9 MAYOR DE 3 NUMEROS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Mayor3Num {

   public static void main(String[] args) {
         //Objeto para lectura de datos
         Scanner leer = new Scanner(System.in);

         byte may, men, num1, num2, num3;
         System.out.print ("Ingrese el primer número: ");
         num1=leer.nextByte();
         System.out.print ("Ingrese el segundo número: ");
         num2=leer.nextByte();
         System.out.print ("Ingrese el tercer número: ");
         num3=leer.nextByte();
       may = num1;
       men = num1;
      if (num2 > may) may = num2;
      if (num3 > may) may = num3;
      if (num2 < men) men = num2;
      if (num3 < men) men = num3;
      System.out.println("Mayor es:" + may);
            System.out.println("Menor es:" + men);

      System.out.println("GRACIAS");

   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa encuentra el mayor y menor número de entre 3 números ingresados por teclado, por ejemplo si se ingresa el 31, 37 y 80, el número mayor sería 80 y el menor sería 31.

TEMA 10 : DESGLOSE DE BILLETES

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Billetes {

   public static void main(String[] args) {
        //Objeto para lectura de datos
                Scanner leer = new Scanner(System.in);

        int can;
                int c100=0, c50=0, c20=0, c10=0, c5=0;
                System.out.print("Ingrese la cantidad a verificar : ");
                can=leer.nextInt();
                if (can >= 100)
                {
                        c100 = (can / 100);
                        can = can - (c100 * 100);
        }

        if (can >= 50)
        {
                c50 = (can / 50);
                can = can - (c50 * 50);
        }
        if (can >= 20)
        {
              c20 = (can / 20);
        can = can - (c20 * 20);
        }

           if (can >= 10)
           {
        c10 = (can / 10);
        can = can - (c10 * 10);
        }

           if (can >= 5)
           {
        c5 = (can / 5);
        can = can - (c5 * 5);
        }
        System.out.println("Billetes de a 100: " + c100);
        System.out.println("Billetes de a 50 : " + c50);
        System.out.println("Billetes de a 20 : " + c20);
```

```
        System.out.println("Billetes de a 10 : " + c10);
        System.out.println("Billetes de a 5  : " + c5);
        System.out.println("Billetes de a 1  : " + can);
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa nos indica la cantidad de billetes de 100, 50, 20, 10, 5 y 1 dólar que se desglosan de un valor ingresado por teclado.

TEMA 11: BONO DEL EMPLEADO POR HIJO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class BonoEmpleado {

   public static void main(String[] args) {
        //Objeto para lectura de datos
                Scanner leer = new Scanner(System.in);

        double sueldo, aRecibir;
                int nHijo, bono;
                String nom;

                System.out.print("Nombre Empleado  : ");
                nom=leer.next();
                System.out.print("Sueldo Empleado  : ");
                sueldo=leer.nextDouble();
                System.out.print("Número de Hijos  : ");
                nHijo=leer.nextInt();

                if(nHijo >= 3)
                        bono= nHijo * 10;
                else
                        bono=nHijo * 20;

                aRecibir = sueldo + bono;

                System.out.println("Recibe : "+ aRecibir);
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza el calculo del bono de un empleado, se calcula según su numero de hijos y se ingresa por teclado su sueldo y número de hijos.

TEMA 12: NÚMERO INTERMEDIO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class NumIntermedio {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);

        int num1, num2, num3;
                num1=50;num2=10;num3=20;
                System.out.print("Primer Número  :");
                num1=leer.nextInt();
                System.out.print("Segundo Número  :");
                num2=leer.nextInt();
                System.out.print("Tercer Número  :");
                num3=leer.nextInt();

                System.out.println();
                System.out.print("El Intermedio es: ");
                if (num1 > num2)
                {
                        if (num1 < num3)
                        System.out.println(num1);
                        else
                        {
                        if (num2 < num3)
                                System.out.println(num3);
                        else
                                System.out.println(num2);
                        }
                }
                else
                {
                if (num2 < num3)
                        System.out.println(num2);
                   else
                   {
                        if (num1 < num3)
                        System.out.println(num3);
                        else
                        System.out.println(num1);
                   }
                }
   }
```

}

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula cual es el número intermedio de entre 3 números ingresados por teclado.

TEMA 13 : TARIFA TELEFÓNICA

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class TarifaTelefonica {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int canKV;
                double  tot, cosKV;
                canKV=1500; cosKV=0;
                System.out.print("Ingrese la cantidad de Kilovatios  :");
                canKV=leer.nextInt();
                if (canKV <= 1000)
                        cosKV = 0.14;
                if ((canKV > 1000) && (canKV <= 1800))
                        cosKV = 0.12;
                if (canKV > 1800)
                        cosKV = 0.8;
                tot = canKV * cosKV;
                System.out.println("Costo de Kilovatio: " + cosKV);
                System.out.println("Total a pagar       : " + tot);
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa calcula el costo de cada kilovatio y el total a pagar ingresando por teclado la cantidad de kilovatios consumidos.

TEMA 14: TRÍANGULOS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Triangulos {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int lado1, lado2, lado3;
                System.out.print("Digite el lado 1 : ");
                lado1=leer.nextInt();
                System.out.print("Digite el lado 2 : ");
                lado2=leer.nextInt();
                System.out.print("Digite el lado 3 : ");
                lado3=leer.nextInt();
                if ((lado1 == lado2) && (lado2 == lado3))
                System.out.println("Triángulo Equilátero. Todos Iguales");
                else
                {
                 if ((lado1 != lado2) && (lado1 != lado3) && (lado2 != lado3))
                   System.out.println("Triángulo Escaleno. Ninguno Igual");
                 else
                   System.out.println("Triángulo Isósceles. Dos Iguales");
                }
                System.out.println("Gracias:");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa nos permite saber si un triángulo es Equilatero, Escaleno o Isósceles, ingresando los 3 lados del triángulo.

TEMA 15: DIA DE LA SEMANA

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class DiaDeSemana {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int dia;
        System.out.print("Ingrese un Número del 1 al 7 :");
        dia=leer.nextInt();
                switch(dia)
                {
                        case 1 :
                                System.out.println("DOMINGO");
                                break;
                        case 2 :
                                System.out.println("LUNES");
                                break;
                        case 3 :
                                System.out.println("MARTES");
                                break;
                        case 4 :
                                System.out.println("MIÉRCOLES");
                                break;
                        case 5 :
                                System.out.println("JUEVES");
                                break;
                        case 6 :
                                System.out.println("VIERNES");
                                break;
                        case 7 :
                                System.out.println("SÁBADO");
                                break;
                        default:
                                System.out.println("Número fuera de Rango");
                        break;
                }
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa nos permite conocer a travez del ingreso de un número del 1 al 7 el nombre del día de la semana.

TEMA 16: ESTADO CIVIL

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class EstadoCivil {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        String eCivil;
                System.out.print("Digite una letra puede ser: C,S,V,D : ");
                eCivil=leer.next();
                eCivil=eCivil.toUpperCase();
                switch (eCivil)
                {
                        case "C" :
                                System.out.println("Casado");
                                break;
                        case "S" :
                                System.out.println("Soltero");
                                break;
                        case "V" :
                                System.out.println("Viudo");
                                break;
                        case "D" :
                                System.out.println("Divorciado");
                                break;
                        default:
                                System.out.println("No existe");
                        break;
                }
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa nos indica el estado civil según una letra ingresado por teclado.

TEMA 17: OPERADORES LÓGICOS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class OperadorLogicos {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
                int i;
                int j;

                //Imprime I y j por consola
                System.out.println ("Ingreso de Datos: ");
                System.out.print("i = " + i);
                i=leer.nextInt();
                System.out.print("j = " + j);
                j=leer.nextInt();
                // Imprime diversas operaciones binarias sobre i y j
                // junto con su resultado.

                System.out.println("i > j es " + (i > j));
                System.out.println("i < j es " + (i < j));
                System.out.println("i >= j es " + (i >= j));
                System.out.println("i <= j es " + (i <= j));
                System.out.println("i == j es " + (i == j));
                System.out.println("i != j es " + (i != j));
                System.out.println("(i < 10) && (j < 10) es: "+ ((i < 10) && (j < 10)) );
                System.out.println("(i < 10) || (j < 10) es: "+ ((i < 10) || (j < 10)) );
                System.out.println();
                System.out.println(" (8 < 10) es " + !( 8< 10));

                System.out.println("Gracias");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El Programa nos indica las diferentes operaciónes lógicas a traves del ingreso de 2 datos, y nos da como resultado un valor binario.

TEMA 18: TABLA DE MULTIPLICAR

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class TablaMultiplicar {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int num,resul,i;
        System.out.print ("Digite un número: ");
                num = leer.nextInt();
                System.out.println ();
                for(i=1;i<=12;i++)
                {
                resul = num * i;
                System.out.println(num+"\t*\t"+i+"\t=\t"+resul);
                }
                System.out.println("\nGRACIAS");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa a traves del ingreso de un número nos permite conocer la tabla de multiplicar de dicho número.

TEMA 19 PRESUPUESTO ANUAL EN ÁREAS HOSPITAL

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class PresupuestoHospital {

   public static void main(String[] args){
                Scanner leer=new Scanner(System.in);
                double canP, tot, porc,can,i;
                char area;

                //#calculos
                System.out.print ("Ingrese la Cantidad de Calculos: ");
                can = leer.nextDouble();
                //TOTAL DEL PRESUPUESTO:
                canP = 150;
                System.out.println("Ginecologia==>: D ");
                System.out.println("Traumatologia==>: T ");
                System.out.println("Pediatra==>: P");
                System.out.println("");

                for(i=1;i<=can;i++)
                {
                        System.out.print("Area==>: ");
                        area=leer.next();

                        switch (area)
                        {
                                case 'G' :
                                        porc = 40;
                                break;
                                case 'T' :
                                        porc = 30;
                                break;
                                case 'P' :
                                        porc = 30;
                                break;
                                default:
                                        porc = 0;
                                break;
                        }
                        tot = (canP * porc) / 100;
                        System.out.println("Su Area recibe " + tot);
                        System.out.println("");
                }
```

```
            System.out.println("Gracias.");
}
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

TEMA 20: SUMA DE N NUMEROS IMPARES E IMPARES

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Suma_n_num_par_impar {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);

        int num, i,sumP= 0,sumI= 0;
                System.out.print("Ingrese un Número: ");
                num=leer.nextInt();

                for(i = 2;i<=num;i+=2)
                {
                        sumP = sumP + i;
                }
                for (i = 1;i<=num;i+=2)
                {
                        sumI = sumI + i;
                }
                System.out.println("Total en Pares : " + sumP);
                System.out.println("Total en Impares : " + sumI);

                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula la suma de los números pares y la suma de los números impares, de los números anteriores de un número ingresado por teclado.

TEMA 21: TABLAS DE MULTIPLICAR

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class TablasDeMultiplicar {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);

        int num, resul, t, i;

                System.out.print("Ingrese el número de Tablas: ");
                num=leer.nextInt();
                for( t = 1; t<=num;t++)
                {
                        System.out.println ();
                        for( i = 10; i>=1;i--)
                        {
                                resul = t * i;
                                System.out.println(t+"\t*\t"+i+"\t=\t"+resul);
                        }
            }
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula un número determinado de tablas de multiplicar; el número de tablas deseado se ingresara por teclado.

TEMA 22: SUMA DE N NÚMEROS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class SumaNNumeros {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int can, k, num,sum= 0;

                System.out.print("Ingrese la cantidad de Números que desee sumar: ");
                can=leer.nextInt();
                for(k =1;k<=can;k++)
                {
                        System.out.print("Digite un número: ");
                        num=leer.nextInt();
                        sum += num;
                }
                System.out.println("Suma total es : " + sum);
                System.out.println("Media Aritmética: " + sum / can);
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula la suma y la media aritmética de n números ingresado por teclados, la cantidad de números a sumarse ingresara por teclado.

TEMA 23: MAYOR Y MENOR DE N NÚMEROS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class MayMen_N_Numeros {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int can, k, may, men, num,dif;

                System.out.print("Ingrese la cantidad de números: ");
                can=leer.nextInt();
                System.out.print("Digite un Número:") ;
                may=leer.nextInt();

                men = may;
                for(k=2;k<=can;k++)
                {
                        System.out.print("Digite un Número:") ;
                        num=leer.nextInt();
                   if(num > may) may = num;
                   if(num < men) men = num;
                }
                dif=may-men;
                System.out.println("El Mayor es : " + may);
                System.out.println("El Menor es : " + men);
                System.out.println("Diferencia es : " + dif);
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa nos permite saber cual es el mayor número, menor número y diferencia entre una cantidad de números ingresados por teclado.

TEMA 24: SERIE DE FIBONACCI

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class SerieFibonacci {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int can, k, a, b, c;

                System.out.print("Números: ");
                can=leer.nextInt();
                a = 1; b = 1;
                System.out.println ();
                System.out.print(a + " " + b + " ");
                for( k = 3 ; k<=can; k++)
                {
                        c = a + b;
                        System.out.print(c + " ");
                        a = b;
                        b = c;
                }
                System.out.println();
                System.out.println("Gracias");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa imprime una cantidad de números determinados de la serie Fibonacci.

TEMA 25: CALIFICACIONES DE UN GRUPO DE ESTUDIANTES

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class CalificacionesEstudiantes {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int can, k;
                double nota1, nota2, prom, sum;
                String nom;

                System.out.print("Cuantos Estudiantes: ");
                can=leer.nextInt();
                sum = 0;
                for(k=1;k<=can;k++)
                {
                        System.out.print("Nombre: ");
                        nom =leer.next();

                        System.out.print("Nota 1: ");
                   nota1=leer.nextInt();

                   System.out.print("Nota 2: ");
                        nota2=leer.nextInt();

                   prom = (nota1 + nota2) / 2;
                        System.out.print("Promedio: " + prom);
                   sum += prom;
                   System.out.println();
                }
                System.out.println("");
                System.out.println("Suma Total es : " + sum);
                System.out.println("Media Aritmética: " + sum / can);
                System.out.println("Gracias.");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa calcula el promedio de 2 notas ingresadas por teclado de cada estudiante, la suma total y media aritmética de todos los estudiantes; la cantidad de estudiantes se ingresa por teclado

TEMA 26: NÚMEROS ALEATORIOS Y CARACTERES ASCII

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
import java.util.Random;
public class Codigo_AscII {

   public static void main(String[] args) {
     Scanner leer=new Scanner(System.in);
     Random rnd=new Random();//Declarar un objeto para la sentencia Random
     int can, k, num;

            System.out.print("Cuantos Números: ");
            can = leer.nextInt();
            System.out.println("");

       //INICIALIZA EL GENERADOR DE ALEATORIOS
            for(k=1;k<=can;k++)
       {
                    num = rnd.nextInt(100);
                    System.out.println("Generó el: " + num);
                    if ((num > 0) && (num < 256))
                    System.out.println("El "+num+" es el código de " + (char)num);
                    System.out.println("");
       }
       System.out.print("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa nos indica a cual código Ascii pertenece cierto número generado aleatoriamente; la cantidad de números se ingresara por teclado.

TEMA 27: COMPARACIÓN DE CADENAS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import javax.swing.JOptionPane;
public class Cadenas {

   public static void main(String[] args) {


      String s1 = new String( "hola" );  // s1 es una copia de "hola"
      String s2 = "adiós";
      String s3 = "Feliz Cumpleaños";
      String s4 = "feliz cumpleaños";

      String salida = "s1 = " + s1 + "\ns2 = " + s2 + "\ns3 = " + s3 +
"\ns4 = " + s4 + "\n\n";

      // probar igualdad
if ( s1.equals( "hola" ) )  // true
         salida += "s1 es igual a \"hola\"\n";
      else
salida += "s1 es distinta de \"hola\"\n";

      // probar igualdad con ==
      if ( s1 == "hola" )  // false; no son el mismo objeto
         salida += "s1 es igual a \"hola\"\n";
      else
         salida += "s1 es distinta de \"hola\"\n";

// probar igualdad (ignorar mayúsculas)
      if ( s3.equalsIgnoreCase( s4 ) )  // true
salida += "s3 es igual a s4\n";
      else
         salida += "s3 es distinta de s4\n";

      // probar compareTo
      salida += "\ns1.compareTo( s2 ) es " + s1.compareTo( s2 ) +
"\ns2.compareTo( s1 ) es " + s2.compareTo( s1 ) +
         "\ns1.compareTo( s1 ) es " + s1.compareTo( s1 ) +
         "\ns3.compareTo( s4 ) es " + s3.compareTo( s4 ) +
         "\ns4.compareTo( s3 ) es " + s4.compareTo( s3 ) + "\n\n";

      // probar regionMatches (susceptible a mayúsculas)
      if ( s3.regionMatches( 0, s4, 0, 5 ) )
salida += "Los primeros 5 caracteres de s3 y s4 concuerdan\n";
```

```
      else
        salida += "Los primeros 5 caracteres de s3 y s4 no concuerdan\n";

// probar regionMatches (ignorar mayúsculas)
      if ( s3.regionMatches( true, 0, s4, 0, 5 ) )
salida += "Los primeros 5 caracteres de s3 y s4 concuerdan";
      else
        salida += "Los primeros 5 caracteres de s3 y s4 no concuerdan";

      JOptionPane.showMessageDialog( null, salida,
        "Comparaciones entre cadenas", JOptionPane.INFORMATION_MESSAGE );

      System.exit( 0 );
}
}

// Referencia http://todojava.awardspace.com/ejemplos-java.html?desc=CompararString
```

3. **GRABAR Y EJECUTAR**

TEMA 28: FUNCIONES DE CADENA

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class FuncionesCadenas {

   public static void main(String[] args) {
     Scanner leer=new Scanner(System.in);

     String nom1,nom2;
            System.out.print ("Digite su nombre: ");
            nom1=leer.next();
            System.out.print ("Digite su Apellido: ");
            nom2=leer.next();
            String nom=nom1+" "+nom2;
            System.out.println("Unido queda : "+nom);
            System.out.println("Mayúsculas : "+nom.toUpperCase());
            System.out.println("Minúsculas : "+nom.toLowerCase());
            System.out.println("Longitud : "+nom.length());
            // Recuerde que la cadena empieza desde la posición 0
            System.out.println("Subcadena : "+nom.substring(0,5));
            System.out.println("Extrae Letra: "+nom.charAt(2));

            System.out.println("GRACIAS.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa nos indica las diferentes funciones de String.

TEMA 29: RELOJ DIGITAL

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class RelojDigital {

   public static void main(String[] args) {
        int h, m, s;
                System.out.println("SIMULACIÓN DE UN RELOJ DIGITAL");
                for(h=0;h<=24;h++)
                {
                        for (m=0;m<=59;m++)
                        {
                                for (s=0;s<=59;s++)
                                {
                                        System.out.println(h+":"+m+":"+s);
                                        try{
                                        Thread.sleep(1000);//Hace una pausa de un segundo
                                        }
                                        catch(InterruptedException e){
                                        }
                                }
                        }
                }

                System.out.print("GRACIAS...");

   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa realiza el funciomiento de un reloj digital.

TEMA 30: CANTIDAD DE VOCALES CERRADAS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Cantidad_VocalesCerradas {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        String nom;
                int k, sum;
                char le;
                System.out.print("Digite una Frase : ");
                nom = leer.next();
                nom=nom.toUpperCase();
                System.out.println (nom);
                sum = 0;
                for(k=0;k<nom.length();k++)
                {
                        le =nom.charAt(k);

                        if(le=='U'|| le=='I')
                                sum = sum + 1;
                }
                System.out.println("Existen "+sum+" Vocales Cerradas.");
                System.out.println("Gracias.");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa nos indica cuantas vocales cerradas existen en una palabra ingresada por teclado.

TEMA 31: ESTADÍSTICA POR VOCAL

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class EstadisticaPorVocal {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        String nom;
                int k, a, e, i, o, u;
                char le;
                System.out.print("Digite una Frase : ");
                nom = leer.next();
                nom=nom.toLowerCase();
                a = 0 ; e = 0 ; i = 0 ; o = 0 ; u = 0;
                for(k=0;k<nom.length();k++)
                {
                        le =nom.charAt(k);
                        switch(le)
                        {
                                case 'a' : a = a + 1; break;
                                case 'e' : e = e + 1;break;
                                case 'i' : i = i + 1;break;
                                case 'o' : o = o + 1;break;
                                case 'u' : u = u + 1;break;
                        }
                }
                System.out.println("Existen "+a+" Vocales A");
                System.out.println("Existen "+e+" Vocales E");
                System.out.println("Existen "+i+" Vocales I");
                System.out.println("Existen "+o+" Vocales O");
                System.out.println("Existen "+u+" Vocales U");
                System.out.println("Gracias.");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa calcula la cantidad de cada vocal que existe en una palabra ingresada por teclado.

TEMA 32: FACTORIAL DE UN NÚMERO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class FactorialNum {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int num, k, resul=1;

            System.out.print("Digite un Número: ");
                num=leer.nextInt();

            for (k = 2 ;k<=num;k++)
                        resul = resul * k;
                System.out.println("El Factorial es: " + resul);
            System.out.println("Gracias.");

   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el factorial de un número ingresado por teclado.

TEMA 33: SERIE DE UN NÚMERO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class SerieDeNum {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        double num, k, f, fac=1,resul = 0;

                System.out.print("Ingrese un número: ") ;
           num=leer.nextInt();

                for (k = 1 ;k<=num;k++)
                {
                        fac=1;
                        for (f = 1;f<=k;f++)
                                fac = fac * f;
                resul = resul + (k / fac);
                }

                System.out.println("Resultado de la serie es: " + resul);
                System.out.println("Gracias.");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa calcula el resultado de la serie de un número ingresado por teclado.

TEMA 34: CAST

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Cast {

   public static void main(String[] args) {

int i = 9,k;
        float j = 47.9F;
        System.out.println("i: "+i + " j: " +j);

        k = (int)j; //Empleo de un cast
        System.out.println("j: " + j + " k: " +k);

        j = k;//No necesita cast
        System.out.println("j: " + j + " k: " +k);

        float m = 2.3F;
        //float m = 2.3; //Daría error al compilar.
        System.out.println("m: "+m);
}
}
```

Es posible convertir un dato de jerarquía "superior" a uno con jerarquía "inferior", arriesgándonos a perder información en el cambio. Este tipo de operación (almacenar el contenido de una variable de jerarquía superior en una de jerarquía inferior) se denomina cast

3. GRABAR Y EJECUTAR

TEMA 35: TABLA DE MULTIPLICAR CON WHILE

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Tabla_multlp_while {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int num, i, resul;

                System.out.print("Ingrese un Número: ");
                num=leer.nextInt();

                i = 1;
                while (i <= 12)
                {
                        resul = num * i;
                        System.out.println(num+"\t*\t"+i+"\t=\t"+ resul);
                        i = i + 1;
                }
                System.out.println("Gracias");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza la tabla de multiplicar con while.

TEMA 36: COMPROBAR SI ES NÚMERO PRIMO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class NumPrimo {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int num, resi, k, sw;

                System.out.print("Ingrese un Número: ");
                num=leer.nextInt();

                k = 2; sw = 0;
                while ((k < num) && (sw == 0))
                        {
                                resi = num % k;
                                if (resi == 0)
                                        sw = 1;
                                else
                                        k = k + 1;
                        }
                if (sw == 0)
                        System.out.println(num + " es número primo.");
                else
                        System.out.println(num + " no es número primo.");
                System.out.println();
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula si un número es primo o no lo es.

TEMA 37: FACTORES PRIMOS DE UN NÚMERO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class FactoresPrimosNum {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int num, resi, k;

           System.out.print("Ingrese un número: ");
                num=leer.nextInt();

           k = 2;
           System.out.println ("Factores Primos: ");
           while (num!=1)
           {
                        resi = num % k;
                        if (resi == 0)
                                {
                                System.out.println(k);
                                num = num / k;
                                }
                        else
                                k = k + 1;
                }
           System.out.println("Gracias...");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula los factores primos de un número ingresado por teclado.

TEMA 38: GENERAR N NÚMEROS PRIMOS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class N_numPrimos {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int num, resi, k, sw, can, x;

              System.out.print("Ingrese la cantidad de Números: ");
          can=leer.nextInt();

          num = 1;
          x = 0;
          System.out.println ("Numeros Primos: ");
          while (x < can)
          {
                    k = 2;
                    sw = 0;

                    while ((k < num) && (sw == 0))
                    {
                              resi = num % k;
                              if (resi == 0)
                                        sw = 1;
                              else
                                        k = k + 1;
                    }
                     if (sw == 0)
                {
                          System.out.println(num + " ");
                          x = x + 1;
                }
                num = num + 1;
          }
          System.out.println("GRACIAS...");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa calcula n cantidad de números primos; la n se ingresara por teclado.

TEMA 39: VERIFICACIÓN DE UNA CLAVE 3 OPORTUNIDADES

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class VerificaionClave {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int k, sw;
                String clave;
                sw = 0 ;
                k = 0;
                do{
                        System.out.print("CLAVE: ");
                        clave=leer.next();
                        clave=clave.toUpperCase();

                        if((clave.compareTo("ARIEL"))==0)
                      sw = 1;
                        else
                                k = k + 1;

                }while((k < 3)&&(sw == 0));

                if(sw == 1)
                                System.out.println("Bienvenido " + clave);
                else
                                System.out.println("Oportunidades Terminadas");

                System.out.println("Gracias");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa compara si la palabra ingresada es igual a la clave, se dan tres oportunidades para ingresarla.

TEMA 40: GENERAR UN NÚMERO ALEATORIO ENTRE 10 Y 30

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class NumAleatorio_10y30 {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int num=0,op;

                do
                {
                        do {
                                double rand=Math.random()*100;
                                num=(int)rand;

                                }while ((num < 10) || (num > 30));

                        System.out.println("SE GENERO EL " + num);
                        System.out.print("Digite 1 para ingresar otro número: ");
                        op=leer.nextInt();

                }while(op==1);
                System.out.println("GRACIAS...");

   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa nos permite generar números aleatorios entre el 10 y el 30.

TEMA 41: JUEGO ADIVINA UN NÚMERO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class AdivinaNum {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int num, adi,i;
                byte sw,opor=4;

                //GENERA EL NÚMERO ENTRE 1 AL 20
                do
          {
                        double rand=Math.random()*100;
                        num = (int)rand;
          }while((num < 1) || (num > 20));

                //PROCESO
          i = 1; sw = 0;
          do
          {
                        System.out.print("Ingresa un Número que pienses que será: ");
                        adi=leer.nextInt();

                        if (adi == num)
                        {
                                System.out.println("Adivinaste!!!");
                                sw = 1;
                        }

                        else
                        {
                                if (adi > num)
                                        System.out.println("Te Pasaste");
                                else
                                        System.out.println("Estás Bajo");
                        }
                    i = i + 1;
          }while ((i <= opor) && (sw == 0));

          if (sw == 0)
                        System.out.println("El número fué: " + num);
                System.out.println();
```

```
            System.out.println("Gracias") ;
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa genera un número aleatorio entre 1 y 20 y tenemos tres intentos para adivinarlo, si no perdemos.

TEMA 42: CONTROL DE UNA FACTURA

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class ControlFactura {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int can, fi,op;
           float prec, tot, paga;
           String prod;

           paga = 0;

           do
           {
                        System.out.print("\nProducto:  ");
                        prod = leer.next();

                        System.out.print("Cantidad:  ");
                        can =leer.nextInt();

                        System.out.print("Precio:    ");
                        prec=leer.nextFloat();

                        tot = can * prec;
                        System.out.print("Total:     "+tot);
                        paga = paga + tot;

                        System.out.println("\n");
                        System.out.print("Digite 1 para ingresar otro producto: ");
                        op=leer.nextInt();
                }while (op== 1);
           System.out.print("\n\nTotal a Pagar :" + paga);
           System.out.println("\nGracias por su compra.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el precio total de cada producto para ello se ingresa el precio y la cantidad de cada producto y calcula el precio final total a pagar; tambien se ingresa el nombre de cada producto.

TEMA 43: VOTACIONES POR SECTOR

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class VotacionesPorSector {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int canN, canS, canC,op;
                String zona;
                canN = 0 ; canC = 0 ; canS = 0;

                do{
                        System.out.print("QUE SECTOR NORTE,CENTRO,SUR: ");
                        zona = leer.next();
                        zona=zona.toUpperCase();
                        if((zona.compareTo("NORTE"))==0)
                                canN = canN + 1;
                        if((zona.compareTo("CENTRO"))==0)
                                canC = canC + 1;
                        if((zona.compareTo("SUR"))==0)
                                canS = canS + 1;

                        System.out.print("\nDigite 1 si hay otra persona: ");
                        op=leer.nextInt();
                        System.out.print("\n");

                }while (op== 1);

                System.out.println("De la zona Norte :" + canN);
                System.out.println("De la zona Centro :" + canC);
                System.out.println("De la zona Sur :" + canS);
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el número de votaciones de cada zona, se ingresa por teclado por cual zona se desea votar.

TEMA 44: PROMEDIO DE SUELDOS CERO O NEGATIVO SALE

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class PromedioSueldos {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        double sumT=0,prom;
                float sue;
                int can=0,op;
                sumT = 0;

                do{

                        System.out.print("Digite Sueldo: ");
                        sue =leer.nextFloat();

                        if (sue > 0)
                        {
                                sumT = sumT + sue;
                                can = can + 1;
                        }
                        System.out.println ("Digite 1 si va a ingresar otro sueldo: ");
                        op=leer.nextInt();
                }while (op==1);
                prom = sumT / can;
                System.out.println();
                System.out.println("Total sueldos: " + sumT);
                System.out.println("Empleados : " + can);
                System.out.println("Promedio : " + prom);
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el promedio la suma y el promedio de todos los sueldos ingresados por teclado.

TEMA 45: FRASE INVERTIDA CON WHILE

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class FraseInvertida_While {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        String nom, aux;
                int pos;
                aux = "";
                System.out.print("Digite una Frase: ");
                nom =leer.next();
                pos =(nom.length())-1;
                while (pos >= 0)
                {
                        aux = aux + nom.charAt(pos);
                        pos = pos - 1;
                }
                System.out.println("Frase  Invertida:" + aux);
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa invierte una palabra usando ciclos while.

TEMA 46: INTERCALACIÓN MAYÚSCULAS Y MINÚSCULAS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Intercalacion_MayusculasMinusculas {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);

        String nom, aux, le;
                int pos;
                aux = "";

                System.out.print("Digite una Frase:");
                nom =leer.next();
                pos = 0;
                while (pos < nom.length())
                {
                        le = nom.substring(pos,pos+1);
                        if (pos % 2 == 0)
                                aux = aux + le.toLowerCase();
                        else
                                aux = aux + le.toUpperCase();
                        pos = pos + 1;
                }
                System.out.println("Frase Intercalada :" + aux);
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa intercala una palabra entre mayúsculas y minúsculas.

TEMA 47: ASIGNACIÓN DIRECTA DE UN CONJUNTO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Serie {

   public static void main(String[] args) {
         String colores = "rojo,amarillo,verde,azul,morado,marrón";

                  int inicio = colores.indexOf(",");
                  int fin = colores.indexOf(",", inicio + 1);

                  System.out.println("Posición de primera coma: "+inicio);
                  System.out.println("Posición de segunda coma: "+fin);
                  System.out.println(colores.substring(inicio + 1, fin));
                  System.out.println(colores.substring(inicio));

                  // OTRO

                  int mesdias[]={31,28,31,30,31,30,31,31,30,31,30,31};
                  System.out.println("Abril " + mesdias[3] + " días");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa nos enseña algunas funciones de String.

TEMA 48: GENERAR NÚMEROS ALEATORIOS EN UN ARREGLO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class NumAleatorios_Arreglo {

   public static void main(String[] args) {
            Scanner leer=new Scanner(System.in);
            int can, pos;
                double[] vec=new double[15];
                int[] vec1=new int [15];

                System.out.print("Cuantos Aleatorios: ") ;
                can = leer.nextInt();

                System.out.print("\n");

                //GENERACIÓN DE ALEATORIOS
                for (pos = 1; pos<=can;pos++)
                {
                        vec[pos] = Math.random();
                        //OBTENEMOS UN ENTERO DE 2 CIFRAS
                        vec1[pos] = (int)(vec[pos] * 100);

                }//SALIDA DEL ARREGLO

                for (pos=1;pos<=can;pos++)
                {
                        System.out.print(vec[pos]);
                        System.out.print("\t\t"+vec1[pos]);
                        System.out.print("\n");
                }

                System.out.println("\nGracias.");

   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO
El programa genera n números aleatorios.

TEMA 49: SUMA ELEMENTOS PARES E IMPARES EN UN ARREGLO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class SumParesImpares_Arreglo {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, can, sumP, sumI;
                int[] vec=new int[12];

                System.out.print("Cuantos Max=12:  ") ;
                can = leer.nextInt();

                //INGRESO
                for(i=1;i<=can;i++)
                {
                        System.out.print("Posición "+ i+"==> :  ");
                        vec[i] = leer.nextInt();
                }

                //PROCESO
                sumP = 0 ; sumI = 0;
                for(i = 1;i<=can;i++)
                {
                        if ((vec[i]%2) == 0)
                                sumP = sumP + vec[i];
                        else
                                sumI = sumI + vec[i];
                }

                //SALIDA
                System.out.println();
                System.out.println("Suma valores Pares: " + sumP);
                System.out.println("Suma valores Impares: " + sumI);
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa suma los valores pares e impares de los elementos ingresados en un vector.

TEMA 50: SUMA POSICIONES PARES E IMPARES EN UN ARREGLO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Sum_Posicions_ParesImpares_Arreglos {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, can,sumP, sumI;
                int[] vec=new int[12];

                System.out.print("Cuantos Elementos max=12: ");
                can = leer.nextInt();
                sumP = 0 ; sumI = 0;
                for (i = 1;i<=can;i++)
                        {
                                System.out.print("Posición "+i+"==> ");
                                 vec[i] = leer.nextInt();

                                 if ((i%2)==0)
                                        sumP = sumP + vec[i];
                                 else
                                        sumI = sumI + vec[i];
                        }

                //SALiDA
                System.out.println();
                System.out.println("Suma valores de posiciones pares : " + sumP);
                System.out.println("Suma valores de posiciones impares: " + sumI);
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula la suma de las posiciones pares y la suma de las posiciones impares de los datos de un vector.

TEMA 51: MAYOR Y MENOR DE UN ARREGLO DE N ELEMENTOS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class NumMayMen_Arreglo {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, can,may, men, pos;

                System.out.print("Ingrese la cantidad de Elementos: ");
                can = leer.nextInt();
                int[] vec= new int[12];

                // iNGRESO
                for(i=1;i<=can;i++)
                {
                        System.out.print("Posición "+i+"==> ");
                        vec[i] = leer.nextInt();
                }

                //PROCESO
                may = vec[1]; pos = 1 ; men = vec[1];
                for(i=2;i<=can;i++)
                {
                        if (vec[i] > may){
                                may = vec[i];
                        pos = i;}

                        if (vec[i] < men)
                                men = vec[i];
                }

                //SALiDA
                System.out.println();
                System.out.println("Mayor es : "+may+" y esta en la posición "+pos);
                System.out.println("El menor es: " + men);
                System.out.println("Gracias.");

   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el mayor y menor número dentro de un vector.

TEMA 52: OBTENER EL DÍGITO VERIFICADOR DE LA CÉDULA

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class DigitoVerificadorCedula {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int sump, sumI, sT, p, aux, dV;
                int[] vec=new int[10];

                // INGRESO
                for (p=1;p<=9;p++)
                {
                        System.out.print("Posición "+p+"==> ");
                        vec[p] = leer.nextInt();
                }

                //SUMATORIA DE PARES
                sump = 0 ; sumI = 0;
                for(p=2;p<=8;p++)
                {
                        sump = sump + vec[p];
                        p++;
                }

                //SUMATORIA DE IMPARES
                for (p=1;p<=9;p++)
                {
                        aux = vec[p]*2;
                        if (aux > 9)
                                aux = aux - 9;
                        sumI = sumI + aux;
                        p++;
                }

                //OBTENCIÓN DE DÍGITO
                sT = sump + sumI;
                dV = 10 - (sT % 10);

                if (dV == 10)
                        dV = 0;
                System.out.println("El dígito verivicador es: " + dV);
                System.out.println("Gracias.");
   }
```

}

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO
El programa calcula el digito verificador de la cédula, ingresando cada dígito de la cédula en un vector.

TEMA 53: INSERTAR UN ELEMENTO EN UN ARREGLO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class InsertarElemento_Arreglo {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, can, pos,ele;

                System.out.print("Cuantos elementos :");
                can = leer.nextInt();

                int[] vec=new int[15];
                int[] nuevo=new int[15];

                // iNGRESO
                for(i=0;i<=can-1;i++)
                {
                        System.out.print("Posición "+i+"==>") ;
                        vec[i] =leer.nextInt();
                }
                System.out.print("Elemento a Insertar: ");
                ele =leer.nextInt();

                do{
                        System.out.print("Posición a insertar: ");
                   pos=leer.nextInt();
                }while ((pos < 1) || (pos > can));

                // PROCESO
                // TRANSLADAMOS DATOS ANTES DE LA posiCiON
                for(i=0;i<pos;i++)
                {
                        nuevo[i] = vec[i];
                }

                //iNSERTAMOS eleMENTO

                nuevo[pos]=ele;

                // TRANSLADAMOS DATOS DESPUES DE LA posiCiON
                for (i=pos;i<=can;i++)
                {
```

```
                nuevo[i + 1] = vec[i];
            }

            // SALiDA
            System.out.println();
            System.out.println("Nuevo Arreglo:");
            for (i=0; i<=can; i++)
                System.out.println(nuevo[i]);

            System.out.println("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El Programa realiza el ingreso de un número en una posicioón en un vector determinado; los datos del vector se ingresara por teclado y se crea un nuevo vector para ingresar el nuevo número.

TEMA 54: ELIMINAR UN ELEMENTO EN UN ARREGLO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class EliminarElemento_Arreglo {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, can, pos;

                System.out.print("Cuantos Elementos :");
                can = leer.nextInt();
                int[] vec=new int[15];
                int[] nuevo=new int[15];

                // iNGRESO
                for(i=1;i<=can;i++)
                {
                        System.out.print("Posición "+i+"==>");
                   vec[i] = leer.nextInt();
                }
                do{
                        System.out.print("Posición a eliminar:");
                   pos = leer.nextInt();

                }while ((pos < 1) || (pos > can));

                // PROCESO
                //TRANSLADAMOS DATOS ANTES DE LA posiCiÓN
                for(i=1;i<pos;i++)
                {
                        nuevo[i] = vec[i];
                }

                //TRANSLADAMOS DATOS DESPUES DE LA posiCiÓN
                for(i=pos+1;i<=can;i++)
                {
                        nuevo[i-1] = vec[i];
                }

                // SALiDA
                System.out.println();
                System.out.println("Nuevo Arreglo:");

                for(i=1;i<=can-1;i++)
```

```
                {
                        System.out.println(nuevo[i]);
                }
                System.out.print("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa elimina un elemento del vector de una posición ingresada por teclado.

TEMA 55: SUMA DE DOS ARREGLOS DE 5 ELEMENTOS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class SumaDe2Arreglos {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i;
                int[] a=new int[6];
                int[] b=new int[6];
                int[] s=new int[6];

                //iNGREsO
                System.out.println("Ingreso Primer y Segundo arreglo");
                for(i=1;i<=5;i++)
                {
                        //aRREGLO 1
                        System.out.print("Pimer arreglo posicion  "+i+": ");
                   a[i]=leer.nextInt();

                        //aRREGLO 2
                   System.out.print("Segundo Arreglo Posición "+i+": ");
                   b[i]=leer.nextInt();

                   s[i] = a[i] + b[i];
                   System.out.println();
                }

                //saLiDa
                System.out.println("Suma de Arreglos");
                for(i=1;i<=5;i++)
                        System.out.println(s[i]);

                System.out.print("Gracias.");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa realiza la suma de cada número de dos vectores y su respuesta se guarda en un tercer vector.

TEMA 56: SUMA DE DOS ARREGLOS DE 5 ELEMENTOS INTERCALADO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class SumaDe2ArreglosIntercalados {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, f;
                int[] a=new int[6];
                int[] b=new int[6];
                int[] s=new int[6];

                //iNGREsO
                System.out.println("Ingreso del Primer y Segundo Arreglo: ");
                for(i=1;i<=5;i++)
                {
                        //aRREGLO 1
                        System.out.print("Primer Arreglo Posición "+i+": ");
                   a[i]=leer.nextInt();

                        //aRREGLO 2
                   System.out.print("Segundo Arreglo Posición "+i+": ");
                   b[i]=leer.nextInt();
                        System.out.println();
                }

                //PROCEsO
                f = 5; //POsiCiÓN DE aRREGLO
                for(i=1;i<=5;i++)
                {
                        s[i] = a[i] + b[f];
                        f = f - 1;
                }

                //saLiDa
                for(i=1;i<=5;i++)
                        System.out.println(s[i]);

                System.out.print("Gracias...");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza la suma intercalada de dos vectores y lo almacena en un tercer vector, osea suma el primero con el último, el segundo con el penúltimo, y asi sucesivamente.

TEMA 57: NÚMERO DECIMAL A BINARIO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class NumeroDecimal_Binario {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);

        int base = 2, num, aux, pos, i ;
                int[] vec=new int[20];

                System.out.print("Digite un Número: ");
                num=leer.nextInt();
                pos = 1;
                while (num >= base)
                {
                        aux = num % base;
                        vec[pos] = aux;
                        pos = pos + 1;
                        num = num / base;
                }
                vec[pos] = num;
                System.out.println();
                //SALiDA
                for(i=pos;i>=1;i--)
                        System.out.print(vec[i]+" ");

                System.out.println();
                System.out.print("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa transforma un número decimal ingresado por teclado en un número binario.

TEMA 58: NÚMERO DECIMAL A OCTAL

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class NumDecimal_Octal {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int base = 8;
                int num, aux, pos, i;
                int[] vec=new int[20];

                System.out.print("Digite un Número: ");
           num=leer.nextInt();
           pos = 1;
           while (num >= base)
           {
                        aux = num % base;
                        vec[pos] = aux;
                        pos = pos + 1;
                        num = num / base;
           }
           vec[pos] = num;
                System.out.println();
           //SALiDA
           for(i=pos;i>=1;i--)
                        System.out.print(vec[i]+" ");

           System.out.println();
           System.out.print("Gracias.");

   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa transforma un número decimal ingresado por teclado en un número Octal.

TEMA 59: NÚMERO DECIMAL A HEXADECIMAL

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Decimal_Hexadecimal {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int base = 16, num, aux, pos, i;
                String[] vec=new String[20];
                String dat;

                System.out.print("Digite un Número:");
                num=leer.nextInt();

      //PROCESO
            pos = 1;
            while (num >= base)
            {
                        aux = num%base;
                        switch(aux)
                        {
                                case 10 :
                                        dat = "A";
                                break;
                                case 11 :
                                        dat = "B";
                                break;
                                case 12 :
                                        dat = "C";
                                break;
                                case 13 :
                                        dat = "D";
                                break;
                                case 14 :
                                        dat = "E";
                                break;
                           case 15 :
                                dat = "F";
                           break;
                                default:
                                        dat = aux+" ";
                                break;
                         }
                        vec[pos] = dat;
```

```
                pos = pos + 1;
                num = num / base;

        }//FiN DEL WHiLE
    switch(num)
        {
            case 10 :
                dat = "A";
            break;
            case 11 :
                dat = "B";
            break;
            case 12 :
                dat = "C";
            break;
            case 13 :
                dat = "D";
            break;
            case 14 :
                dat = "E";
            break;
          case 15 :
            dat = "F";
          break;
          default:
            dat = num+" ";
          break;
        }
        vec[pos] = dat;

        //SALiDA
        for(i=pos;i>=1;i--)
            System.out.print(vec[i] + " ");

        System.out.println();
        System.out.print("Gracias.");

  }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa transforma un número decimal ingresado por teclado en un número Hexadecimal.

TEMA 60: ORDENAMIENTO DE UN ARREGLO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class OrdenamientoArreglo {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, k, can, aux;

                System.out.print("Cuantos elementos max=12: ");
                can=leer.nextInt();

                int[] vec=new int[13];

                //iNGRESO
                for(i=1;i<=can;i++)
                {
                        System.out.print("Posición "+i+"==> ");
                   vec[i]=leer.nextInt();
                }

                // PROCESO
                for(i=1;i<=can;i++)
                {
                        for(k=i;k<=can;k++)
                        {
                                if (vec[k] > vec[i])
                                {
                                        aux = vec[k];
                                        vec[k] = vec[i];
                                        vec[i] = aux;
                                }
                        }
                }

                System.out.println();

                //ARREGLO
                System.out.println("Arreglo Ordenado Descendentemente:");
                for(i=1;i<=can;i++)
                        System.out.println(vec[i]);

                System.out.print("Gracias.");
   }
```

}

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO
El programa realiza el ordenamiento descendentemente de un vector.

TEMA 61: ORDENAMIENTO DE UN ARREGLO MÉTODO BURBUJA

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class OrdenamientoArregloMetodoBurbuja {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, k, can, aux;

                System.out.print("CUANTOS ELEMENTOS MÁX=12:");
                can=leer.nextInt();

                int[] vec=new int[13];

                //iNGRESO
                for(i=1;i<=can;i++)
                {
                        System.out.print("Posición "+i+"==> ");
                   vec[i]=leer.nextInt();
                }

                //PROCESO
                for(i=1;i<can;i++)
                {
                        for(k=i+1;k<=can;k++)
                        {
                                if (vec[i] < vec[k])
                                {
                                        aux = vec[k];
                                        vec[k] = vec[i];
                                        vec[i] = aux;
                                }
                        }
                }
                System.out.println("");

                //REM SALiDA
                System.out.println("Arreglo Ordenado Descendente");
                for(i=1;i<=can;i++)
                        System.out.println(vec[i]);
                System.out.print("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa ordena el vector por el metodo de ordenamiento burbuja.

TEMA 62: BÚSQUEDA DE UN ELEMENTO EN UN ARREGLO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class BusquedaElemento_Arreglo {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, can,tot, bus;

                System.out.print("Cuantos elementos: ");
                can=leer.nextInt();

                int[] vec= new int[13];

                // iNGRESO
                for(i=1;i<=can;i++)
                {
                        System.out.print("Posición "+i+"==> ");
                   vec[i]=leer.nextInt();

                }

                System.out.print("Elemento a buscar: ") ;
                bus=leer.nextInt();

                // PROCESO
                tot = 0;
                for(i=1;i<=can;i++)
                {
                        if (vec[i]==bus)
                                tot = tot + 1;
                }

                //SALiDA
                System.out.println("");
                System.out.println("Existe "+tot+" números "+bus);
                System.out.print("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza la búsqueda de un número en un vector.

TEMA 63: BÚSQUEDA BINARIA DE UN ELEMENTO EN UN ARREGLO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class BusquedaBinaria_ElementoArreglo {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, can, j, alto, bajo, central=0,bus, aux;

                System.out.print("Cuantos Elementos: ");
                can=leer.nextInt();

                int[] vec=new int[15];

                boolean  encontrado=false;

                // iNGRESO
                for(i=1;i<=can;i++)
                        vec[i] =(int)(Math.random()* 100);

                //SALiDA DEL ARREGLO ALEATORiO
                System.out.println ("\nVector : ");
                for(i=1;i<=can;i++)
                        System.out.println(vec[i]);

                //PRiMERO ORDENAMOS EL ARREGLO
                for(j=1;j<=can;j++)
                {
                        for(i=1;i<can;i++)
                        {
                                if (vec[i]>vec[i+1])
                                {
                                        aux = vec[i];
                                        vec[i]=vec[i+1];
                                        vec[i+1]=aux;
                                }
                        }
                }

                //SALiDA DEL ARREGLO ORDENADO
                System.out.println("Arreglo Ordenado");
                for(i=1;i<=can;i++)
                        System.out.println(vec[i]);

                //AHORA Si LA BÚSQUEDA
```

```
            System.out.print("Elemento a Buscar: ");
            bus=leer.nextInt();

            bajo = 1;
            alto = can;

            //central = (bajo + alto) / 2
            while ((bajo <= alto) && (encontrado == false))
            {
                central=(bajo+alto)/2;
                if (vec[central]==bus)
                    encontrado = true;
                else
                {
                    if (vec[central] > bus)
                        alto = central - 1;
                    else
                        bajo = central + 1;
                }
            }

            if (encontrado)
                System.out.println(bus+" encontrado en la posicion "+central);
            else
                System.out.println("No existe "+bus);

            System.out.print("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza una búsqueda binaria en un vector de n cuyos elementos fueron generados aleatoriamente.

TEMA 64: TABLAS DE MULTIPLICAR EN UNA MATRIZ DE NxM

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class TablasMultiplicar_MatrizNxN {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n;

                // GEnERO LOS nÚmEROS
                System.out.print("Número de Filas y Colunmas de la Matriz: ");
                n=leer.nextInt();

                int[][] mat=new int[n+1][n+1];

                for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                                mat[f][c]= f * c;
                }
                System.out.println ();
                //SALIDA
                for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                                System.out.print(mat[f][c]+"\t");
                        System.out.println("");
                }
                System.out.print("");
                System.out.print("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza el calculo de una tabla de multiplicar en una matriz de NxN; la n se ingresara por teclado.

TEMA 65: GENERAR ALEATORIOS EN UNA MATRIZ DE 5x5

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Aleatorios_Matriz5x5 {

   public static void main(String[] args) {
        int f, c;
                int[][] mat=new int[6][6];

                //GENERO LOS NÚMEROS
                for(f=1;f<=5;f++)
                {
                        for(c=1;c<=5;c++)
                                mat[f][c] =(int)(Math.random()* 100);
                }

                //SALIDA
                System.out.println ("Matriz 5x5");
                for(f=1;f<=5;f++)
                {
                        for(c=1;c<=5;c++)
                           System.out.print(mat[f][c]+"\t");
                        System.out.println("");
                }
                System.out.println("");
                System.out.print("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa genera una matriz con números aleatorios con una dimensión de 5x5.

TEMA 66: SUMAR ELEMENTOS DE UNA MATRIZ DE NxN

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Suma_ElementosMatrizNxN {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n, sum= 0;

                // InGRESO
                System.out.print("Tamaño de la Matriz: ");
                n=leer.nextInt();

                int[ ][ ] mat=new int[n+1][n+1];
                for(f=1;f<=n;f++)
                {
                 System.out.println("Fila "+f+":");
                        for(c=1;c<=n;c++)
                        {
                        System.out.print("\t");
                                mat[f][c]=leer.nextInt();
                        }
                        System.out.println("");
                }

                //PROcESO
                for(f=1;f<=n;f++)
                        {
                                for(c=1;c<=n;c++)
                                        sum = sum + mat[f][c];
                        }

                //SALIDA
                System.out.println("");
                System.out.println("Suma total es:" + sum);
                System.out.println("Promedio total es:" + sum/(n * n));
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula la suma total de una matriz ingresada por teclado de dimensión NxN.

TEMA 67: SUMAR ELEMENTOS DE FILA Y UNA COLUMNA MATRIZ DE 5x5

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class SumaElementos_FilaColumna_Matriz5x5 {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, sumF, sumC ;
                //INGRESO
                int[][] mat=new int[6][6];

                for(f=1;f<=5;f++)
                {
                 System.out.println("Fila "+f+" :  ");
                        for(c=1;c<=5;c++)
                        {
                        System.out.print(" ");
                                mat[f][c]=leer.nextInt();
                        }
                        System.out.println("");
                }

                //fILA 2
                sumF = 0;
                for(c=1;c<=5;c++)
                        sumF = sumF + mat[2][c];

                //cOLUMNA 3
                sumC = 0;
                for(f=1;f<=5;f++)
                        sumC = sumC + mat[f][3];

                //SALIDA
                System.out.println("");
                System.out.println("Suma Fila 2 es:" + sumF);
                System.out.println("Suma Columna 3 es:" + sumC);
                System.out.println("Gracias.");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa realiza la suma de la fila 2 y de la columna 3, con los datos de la matriz ingresados por teclado.

TEMA 68: SUMAR ELEMENTOS DE DIAGONAL PRINCIPAL

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class SumaDiagonalPrincipal {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n, sumP, sumS ;

                //InGRESO
                //PARA cOMODIDAD GEnERAMOS VALORES PARA LA matRIZ
                System.out.print("Tamaño de la Matriz: ");
                n=leer.nextInt();

                int[][] mat= new int[n+1][n+1];
                for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                mat[f][c]=(int)(math.random()* 10);
                                System.out.print(mat[f][c]+"  ");
                   }
                   System.out.println("");
                }

                //DIAGOnAL PRIncIPAL
                sumP = 0;
                for(f=1;f<=n;f++)
                        sumP = sumP + mat[f][f];

                //SALIDA
                System.out.println("");
                System.out.println("Suma Diagonal Principal es:" + sumP);
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula la suma de la diagonal principal de una matriz de 5x5, sus datos son generados aleatoriamente.

TEMA 69: SUMAR ELEMENTOS DE DIAGONAL PRINCIPAL Y SECUNDARIAMATRIZ DE NxN

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class SumaDiagonal_PrincipalSecundaria {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n, sumP, sumS ;

                //InGRESO
                //PARA cOMODIDAD GEnERAMOS VALORES PARA LA matRIZ
                System.out.print("Tamaño de la Matriz:");
                n=leer.nextInt();

                int[][] mat= new int[n+1][n+1];
                for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                mat[f][c]=(int)(Math.random()* 10);
                                System.out.print(mat[f][c]+"  ");
                   }
                   System.out.println("");
                }

                //DIAGOnAL PRIncIPAL
                sumP = 0;
                for(f=1;f<=n;f++)
                        sumP = sumP + mat[f][f];

                //DIAGOnAL SEcUnDARIA
                sumS = 0 ; c = n;
                for(f=1;f<=n;f++)
                {
                        sumS = sumS + mat[f][c];
                        c = c - 1;
                }

                //SALIDA
                System.out.println("");
                System.out.println("Suma Diagonal Principal es :" + sumP);
                System.out.println("Suma Diagonal Secundaria es:" + sumS);
                System.out.println("Gracias.");
   }
```

}

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza la suma de la diagonal principal y secundaria de una matriz, sus datos son generados aleatoriamente.

TEMA 70: NÚMERO MAYOR Y MENOR EN UNA MATRIZ DE NxN

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class NumMayMen_MatrizNxN {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n, may, men;

                //InGRESO
                //PARA cOMODIDAD GEnERAMOS VALORES PARA LA matRIZ
                System.out.print("Tamaño de la matriz: ");
                n=leer.nextInt();

                int[][] mat=new int[15][15];

                for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                mat[f][c]=(int)(Math.random()* 10);
                                System.out.print(mat[f][c]+"  ");
                   }
                   System.out.println("");
                }

                //PROcESO
                may = mat[1][1];
                men = mat[1][1];

                for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                if (mat[f][c]> may)
                                        may = mat[f][c];
                                if (mat[f][c] < men)
                                        men = mat[f][c];
                        }
                }
                //SALIDA
                System.out.println("");
                System.out.println("Número Mayor es: " + may);
                System.out.println("Número Menor es: " + men);
```

```
            System.out.println("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el número mayor y menor de una matriz de NxN generada aleatoriamente.

TEMA 71: ORDENAMIENTO DE UNA MATRIZ DE NxN

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Ordenamiento_MatrizNxN {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, i, k, n, aux;

                //inGRESO
                //PARA cOMODiDAD GEnERAMOS VALORES PARA LA matRiZ
                System.out.print("Tamaño de la matriz: ");
                n=leer.nextInt();

                int[][] mat=new int[n+1][n+1];

                for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                mat[f][c]=(int)(Math.random()* 10);
                                System.out.print(mat[f][c]+"  ");
                   }
                   System.out.println("");
                }

                //PROcESO
                for (f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                for(i=1;i<=n;i++)
                                {
                                        for(k=1;k<=n;k++)
                                        {
                                                if (mat[f][c]< mat[i][k])
                                                {
                                                        aux = mat[f][c];
                                                        mat[f][c] = mat[i][k];
                                                        mat[i][k] = aux;
                                                }
                                        }
                                }
                        }
```

```
                }
                //SALiDA
                System.out.println("\nMatriz Ordenada");
                for (f=1;f<=n;f++)
                {
                        for (c=1;c<=n;c++)
                        {
                                System.out.print(mat[f][c]+"  ");
                        }
                        System.out.println("");
                }

                System.out.println("");
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza el ordenamiento de una matriz de NxN; sus datos son generados aleatoriamente.

TEMA 72: SUMA DE MATRICES DE 5x5

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class SumaMatrices5x5 {

   public static void main(String[] args) {
        int f, c;
                int[][] matA=new int[6][6];
                int[][] matB=new int[6][6];
                int[][] matC=new int[6][6];

                //INGRESO
        //PARA COMODIDAD GENERAMOS VALORES PARA LAS MATRICES
                for(f=1;f<=5;f++)
                {
                        for(c=1;c<=5;c++)
                        {
                                matA[f][c]=(int)(Math.random()* 10);
                                matB[f][c]=(int)(Math.random()* 10);
                }
                }

                //SALIDA MATRIZ A
                System.out.println("Matriz A: ");
                for(f=1;f<=5;f++)
                {
                        for(c=1;c<=5;c++)
                        {
                                matA[f][c]=(int)(Math.random()* 10);
                                System.out.print(matA[f][c]+"  ");
                        }
                     System.out.println("");
                }

                //SALIDA MATRIZ B
                System.out.println("Matriz B: ");
                for(f=1;f<=5;f++)
                {
                        for(c=1;c<=5;c++)
                        {
                                matB[f][c]=(int)(Math.random()* 10);
                                System.out.print(matB[f][c]+"  ");
                   }
                      System.out.println("");
                }
```

```
                //PROcESO
                for(f=1;f<=5;f++)
                {
                        for(c=1;c<=5;c++)
                        {
                                matC[f][c]=matA[f][c]+matB[f][c];
                        }
                }

                //SALIDA
                System.out.println("");
                System.out.println("Matriz Resultante:  ");
                for(f=1;f<=5;f++)
                {
                        for(c=1;c<=5;c++)
                        {
                                System.out.print(matC[f][c]+"\t");
                   }
                      System.out.println("");
                }
                System.out.println("");
                System.out.print("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza la suma de 2 matrices de 5x5 generadas aleatoriamente, la suma se guarda en una nueva matriz.

TEMA 73: MULTIPLICACIÓN DE MATRICES DE 4x4

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class MultiplicacionMatrices {

   public static void main(String[] args) {
        int n = 4; //TAMAÑO DE LA MATRIZ
                int  f, c, k;
                int[][] matA=new int[5][5];
                int[][] matB=new int[5][5];
                int[][] matC=new int[5][5];

                //SALIDA MATRIZ A
                System.out.println("Matriz A: ");
                for(f=1;f<=4;f++)
                {
                        for(c=1;c<=4;c++)
                        {
                                matA[f][c]=(int)(Math.random()* 10);
                                System.out.print(matA[f][c]+"  ");
                }
                      System.out.println("");
                }

                //SALIDA MATRIZ B
                System.out.println("MATRIZ B: ");
                for(f=1;f<=4;f++)
                {
                        for(c=1;c<=4;c++)
                        {
                                matB[f][c]=(int)(Math.random()* 10);
                                System.out.print(matB[f][c]+"  ");
                   }
                      System.out.println("");
                }

                //PROCESO
                for(f=1;f<=4;f++)
                {
                        for(c=1;c<=4;c++)
                        {
                                for(k=1;k<=4;k++)
                                        matC[f][c]=matC[f][c] + (matA[f][k] * matB[k][c]);
                        }
```

```
                }

                //SALIDA
                System.out.println("");
                System.out.println("Matriz Resultante:  ");
                for(f=1;f<=4;f++)
                {
                        for(c=1;c<=4;c++)
                        {
                                System.out.print(matC[f][c]+"\t");
                   }
                      System.out.println("");
                }
                System.out.println("");
                System.out.print("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza la multiplicación de 2 matrices de 5x5 generadas aleatoriamente, la multiplicación se guarda en una nueva matriz.

TEMA 74: GENERACIÓN DEL TRIÁNGULO DE PASCAL FORMA 1

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class TrianguloPascal_Forma1 {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n;

                System.out.print("Cuantas Filas: ");
                n=leer.nextInt();

                int[][]mat=new int[n+1][n+1];
                //PROcESO
                for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                if ((c==1)||(f==c))
                                        mat[f][c]=1;
                                else
                                        mat[f][c]= mat[f-1][c] + mat[f-1][c-1];
                        }
                }
                //SALIDA
                for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                if (mat[f][c]!=0)
                                        System.out.print(mat[f][c]+(" "));
                        }
                System.out.println("");
                }
                System.out.println("Gracias.");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa calcula e imprime el triángulo de pascal, se ingresa por teclado el número de filas que deseemos visualizar.

TEMA 75: GENERACIÓN DEL TRIÁNGULO DE PASCAL FORMA 2

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class TrianguloPascal_Forma2 {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n,col, fi;

                System.out.print("Cuantas Filas: ");
                n=leer.nextInt();
                int[][]mat=new int[n+1][n+1];

                //PROCESO
                for(f=1;f<= n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                if ((c == 1) || (f == c))
                                        mat[f][c] = 1;
                                else
                                        mat[f][c] = mat[f - 1][c] + mat[f - 1][c - 1];
                        }
                }
                //SALIDA
                fi = 2;
                for(f=1;f<=n;f++)
                {
                        col = 40 - (f * 2);
                        for(c=1;c<=n;c++)
                        {
                                if (mat[f][c] != 0)
                                {
                                        System.out.print(mat[f][c]+(" "));
                                        col = col + 4;
                                }
                        }
                        fi = fi + 1;
                        System.out.println("");
                }
                System.out.print("Gracias.");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa imprime el triángulo de Pascal.

TEMA 76: MATRIZ TRANSPUESTA DE NxN

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class MatrizTranspuestaNxN {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n ;

                //INGRESO
                //PARA COMODIDAD GENERAMOS VALORES PARA LA MATRIZ
                System.out.print("Tamaño de la Matriz: ");
                n=leer.nextInt();

                int[][] mat=new int[n+1][n+1];
                int[][] matT=new int[n+1][n+1];

                for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                mat[f][c]=(int)(Math.random()* 10);
                                System.out.print(mat[f][c]+"\t");
                   }
                   System.out.println("");
                }

                //PROcESO
                for(f=1;f<=n;f++)
           {
                        for(c=1;c<=n;c++)
                        matT[f][c]=mat[c][f];
                }

                //SALIDA
                System.out.println("\nMatriz Transpuesta: ");
                for (f=1;f<=n;f++)
                {
                        for (c=1;c<=n;c++)
                        {
                                System.out.print(matT[f][c]+"  ");
                        }
                        System.out.println("");
                }
```

```
                System.out.println("");
                System.out.println("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza la matriz transpuesta de una matriz de NxN generada aleatoriamente.

TEMA 77: MAYORES DE CADA FILA DE UNA MATRIZ NxN EN UN VECTOR

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class MayFila_MatrizNxN {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n, may;

                //INGRESO
        //PARA COMODIDAD GENERAMOS VALORES PARA LA MATRIZ
           System.out.print("Tamaño de la Matriz: ");
                n=leer.nextInt();

                int[][] mat=new int[15][15];
                int[] vec=new int[15];

           for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                mat[f][c]=(int)(Math.random()* 10);
                                System.out.print(mat[f][c]+"  ");
                   }
                   System.out.println("");
                }

           //PROCESO
           for(f=1;f<=n;f++)
           {
                        may = mat[f][1];
                        for(c=1;c<=n;c++)
                        {
                                if(mat[f][c] > may)
                                        may = mat[f][c];
                        }
                        vec[f] = may;
                }

           //SALIDA
           System.out.println("Los Mayores de cada fila: ");
                for(f=1;f<=n;f++)
                        System.out.println(vec[f]+"  ");
           System.out.println("");
```

```
            System.out.println("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el número mayor de cada fila de una matriz, cada número se guarda en un nuevo vector.

TEMA 78: MENORES DE CADA COLUMNA DE UNA MATRIZ NxN EN UN VECTOR

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class MenColumna_MatrizNxN {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
                int f, c, n, men;

                //INGRESO
           //PARA COMODIDAD GENERAMOS VALORES PARA LA MATRIZ
           System.out.print("Tamaño de la Matriz: ");
                n=leer.nextInt();

                int[][] mat=new int[15][15];
                int[] vec=new int[15];

           for(f=1;f<=n;f++)
                {
                        for(c=1;c<=n;c++)
                        {
                                mat[f][c]=(int)(Math.random()* 10);
                                System.out.print(mat[f][c]+"  ");
                   }
                   System.out.println("");
                }

           //PROCESO
           for(c=1;c<=n;c++)
           {
                men = mat[1][c];
                for(f=1;f<=n;f++)
                {
                        if (mat[f][c]< men)
                                men = mat[f][c];
                }
                vec[c] = men;
        }

            //SALIDA
           System.out.println("Los Menores de cada Fila: ");
                for(f=1;f<=n;f++)
                        System.out.print(vec[f]+"  ");
```

```
            System.out.println("");
            System.out.println("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el menor número de cada fila de una matriz.

TEMA 79: PROMEDIOS DE CADA COLUMNA DE UNA MATRIZ NxN EN UNVECTOR

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class PromediosColumna_MatrizNxN {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n, sum;

              //INGRESO
          //PARA COMODIDAD GENERAMOS VALORES PARA LA MATRIZ

          System.out.print("Tamaño de la Matriz: ");
              n=leer.nextInt();

              int[][] mat=new int[15][15];
              double [] vec=new double[15];

          for(f=1;f<=n;f++)
              {
                      for(c=1;c<=n;c++)
                      {
                              mat[f][c]=(int)(Math.random()* 10);
                              System.out.print(mat[f][c]+"  ");
                 }
                 System.out.println("");
              }

          //PROCESO
          for(c=1;c<=n;c++)
          {
                      sum = 0;
                      for(f=1;f<=n;f++)
                              sum = sum + mat[f][c];
                      vec[c] = (float) sum / n;
              }

          //SALIDA
          System.out.println("\nPromedios: ");
          for(c=1;c<=n;c++)
          {
              System.out.println("Promedio Columna "+c+": "+vec[c]);
          }
```

```
            System.out.println("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el promedio de cada columna de una matriz de NxN.

TEMA 80: VOTACIONES. SUMA DE CADA COLUMNA REPRESENTA A UN CANDIDATO, OBTENER EL CANDIDATO GANADOR

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class Votaciones_SumaColumnas {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int f, c, n, col, may, sum;

              //INGRESO
          //PARA COMODIDAD GENERAMOS VALORES PARA LA MATRIZ

          System.out.print("Tamaño de la Matriz: ");
              n=leer.nextInt();

              int[][] mat=new int[15][15];
              int [] vec=new int[15];

          for(f=1;f<=n;f++)
              {
                      for(c=1;c<=n;c++)
                      {
                              mat[f][c]=(int)(Math.random()* 10);
                              System.out.print(mat[f][c]+"  ");
                 }
                 System.out.println("");
              }

          //PROCESO
          for(c=1;c<=n;c++)
          {
                      sum = 0;
                      for(f=1;f<=n;f++)
                              sum = sum + mat[f][c];
                      vec[c] = sum;
          }

          may = vec[1];
          col = 1;
          for(c=2;c<=n;c++)
          {
                      if (vec[c] > may){
                              may = vec[c];
```

```
                        col = c;
                    }
            }

            //SALIDA
            System.out.println("***RESULTADOS***");
                for(c=1;c<=n;c++)
                        System.out.print(vec[c]+"  ");
            System.out.println("");
                System.out.print("Número mayor es "+may+" está en la columna "+col);
                System.out.println("");
            System.out.print("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula la suma de cada columna, la almacena en un vector y compara cual es la mayor suma de las columnas.

TEMA 81: ORDENAR CADA FILA DE UNA MATRIZ NxN

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Scanner;
public class OrdenarFila_MatrizNxN {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        int i, f, c, n, aux;

              //INGRESO
           //PARA COMODIDAD GENERAMOS VALORES PARA LA MATRIZ
           System.out.print("Tamaño de la Matriz: ");
              n=leer.nextInt();

              int[][] mat=new int[15][15];
              int [] vec=new int[15];

           for(f=1;f<=n;f++)
              {
                    for(c=1;c<=n;c++)
                    {
                          mat[f][c]=(int)(Math.random()* 10);
                          System.out.print(mat[f][c]+"  ");
                 }
                 System.out.println("");
              }

           //PROCESO
           for(f=1;f<=n;f++)
           {
              for(c=1;c<=n;c++)
              {
                    for(i=1;i<=n;i++)
                    {
                          if (mat[f][c] < mat[f][i])
                          {
                                aux = mat[f][c];
                                mat[f][c] = mat[f][i];
                                mat[f][i] = aux;
                          }
                    }
              }
           }
```

```
            //SALiDA
            System.out.println("\nMatriz Ordenada");
            for (f=1;f<=n;f++)
            {
                        for (c=1;c<=n;c++)
                        {
                                    System.out.print(mat[f][c]+"  ");
                        }
                  System.out.println("");
            }

                  System.out.println("");
                  System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa ordena cada fila de una matriz de NxN.

TEMA 82: CUBO DE UN NÚMERO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
class Cubo{

        public int cubo(int aux){
                aux=aux*aux*aux;
                return aux;
        }
 }
import java.util.Scanner;
public class CuboDeUnNumero {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        Cubo objeto=new Cubo();
                int num, resul;

                System.out.print("Digite un Número: ");
           num=leer.nextInt();

                resul =objeto.cubo(num);
           System.out.print("El cubo de "+num+" es: "+resul);
                System.out.println("");
           System.out.println("Gracias.");

   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza el calculo de un número elevado al cubo, usando clases.

TEMA 83: MAYOR DE TRES NÚMEROS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Mayor{
        public int mayor(int n1,int n2, int n3){
                int may;
                may = n1;
                if (n2 > may) may = n2;
                if (n3 > may)  may = n3;
                return may;
        }
 }
import java.util.Scanner;
public class May3Num_Clases {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        Mayor objeto=new Mayor();
           int num1, num2, num3,resul;

           System.out.print("Digite número 1: ");
           num1=leer.nextInt();

                System.out.print("Digite número 2: ");
                num2=leer.nextInt();

                System.out.print("Digite número 3: ");
                num3=leer.nextInt();

           resul = objeto.mayor(num1, num2, num3);
           System.out.println("\nEl mayor es: " + resul);
           System.out.println("");
           System.out.println("Gracias.");
   }
}
```

3. GRABAR Y EJECUTAR

FUNCIONAMIENTO

El programa realiza el calculo de un número mayor entre 3 números ingresados por teclado.

TEMA 84: VALOR ABSOLUTO DE UN NÚMERO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class MiAbs {
        public int Absoluto(int aux){
                if (aux < 0)
                        aux = aux * -1;
                return aux;
        }
}
import java.util.Scanner;
public class ValorAbsolutoNum_Clase {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        MiAbs objeto=new MiAbs();
                int num, resul;

                System.out.print("Digite un Número: ");
           num=leer.nextInt();

                resul =objeto.Absoluto(num);
           System.out.println("Valor Absoluto es: " + resul);
                System.out.println("");
           System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el valor absoluto de un número ingresado por teclado.

TEMA 85: FACTORIAL DE UN NÚMERO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Factorial {
        public int fact(int aux){
                int res=1, k;
                for(k=2;k<=aux;k++)
                        res = res * k;
                return res;
        }
}
import java.util.Scanner;
public class FactorialNum_Clase {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        Factorial objeto=new Factorial();
                int num, resul;

                System.out.print("DIGITE UN NÚMERO: ");
           num=Integer.valueOf(linea);

                resul =objeto.Factorial(num);
           System.out.println("EL Factorial DE "+num+" ES: "+ resul);
                System.out.println("");
           System.out.println("GRACIAS...");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el factorial de un número ingresado por teclado.

TEMA 86: INVERTIR UNA FRASE CLASES

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Invertir {
        public String inver (String frase){
                String aux;
                int pos;
                aux = "";
                for(pos=(frase.length())-1;pos>=0;pos--)
                        aux = aux + frase.charAt(pos);
                return aux;

        }
}
import java.util.Scanner;
public class InvertirFrase_Clases {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        Invertir objeto=new Invertir();
                String nom,r;
                System.out.print("Digite una Frase:");
                nom =leer.next();
                r=objeto.inver(nom);
                System.out.print("Resultado es: " + r);
                System.out.println("");
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa invierte una palabra ingresada por teclado, para ello se usara clases externas.

TEMA 87: COMPROBAR SI UN NÚMERO ES MÚLTIPLO DE OTRO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Multiplo {
        public String calcular(int n1, int n2){
        String r;
        if (n1 % n2 == 0)
                r = " es Múltiplo ";
        else
                r = " no es Múltiplo ";

       return r;
       }

}
import java.util.Scanner;
public class NumMultiploOtro_Clase {

   public static void main(String[] args) {
       Scanner leer=new Scanner(System.in);
       Multiplo objeto=new Multiplo();
               int num1, num2;
               String resul;

               System.out.print("Digite número 1: ");
          num1=leer.nextInt();

          System.out.print("Digite número 2: ");
          num2=leer.nextInt();

          resul = objeto.calcular(num1,num2);
          System.out.println(num1+resul+num2);
               System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa ingresa 2 números por teclado y compara si el primer número es múltiplo del segundo.

TEMA 88: NÚMERO A QUE DÍA DE LA SEMANA CORRESPONDE

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class DiaSemana {
        public String Dia (int num){
                String r;
                switch(num){
                        case 1 :
                                r = "DOMINGO";
                        break;
                        case 2 :
                                r = "LUNES";
                        break;
                        case 3 :
                                r = "MARTES";
                        break;
                        case 4 :
                                r = "MIÉRCOLES";
                        break;
                        case 5 :
                                r = "JUEVES";
                        break;
                        case 6 :
                                r = "VIERNES";
                        break;
                        case 7 :
                                r = "SÁBADO";
                        break;
                        default:
                                r = "FUERA DE RANGO";
                        break;
                }
                return r;
        }
}
import java.util.Scanner;
public class DiaDeLaSemana_Clase {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        DiaSemana objeto =new DiaSemana();
           int num;
           String resul;

           System.out.print("Digite un número entre 1 y 7: ");
```

```
        num=leer.nextInt();

                resul = objeto.Dia(num);
                System.out.println("");
                System.out.println("Resultado es: " + resul);
                System.out.print("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa indica que día de la semana es al ingresar un número entre 1 al 7 por teclado.

TEMA 89: NÚMERO COMPROBAR SI ES PRIMO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
class NumPrimo {
        public String Primo(int n ){
                int k, sw, resi;
                k = 2 ; sw = 0;
                while ((k < n)&& (sw == 0))
                {
                        resi = n % k;
                        if (resi==0)
                                sw = 1;
                        else
                                k = k + 1;
                }

                if (sw==0)
                        return "Primo";
                else
                        return "No Primo";
        }
}
import java.util.Scanner;
public class NumPrimo_Clase {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        NumPrimo objeto=new NumPrimo();
           int num;
           String resul;

           System.out.print("Digite un Número : ");
           num=leer.nextInt();

                resul =objeto.Primo(num);
           System.out.println("");
           System.out.println("Resultado es: " + resul);
           System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

SOLUCIONADO

El programa nos indica si un número ingresado por teclado es

TEMA 90: MENOR EN UN ARREGLO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class VectorMenor {
        public int menor(int[] nuevo, int n){
                int men, i;
                men = nuevo[1];
                for(i=2;i<=n;i++)
                        if (nuevo[i] < men) men = nuevo[i];
                return men;
        }
}
import java.util.Scanner;
public class MenorEnArreglo_Clase {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        VectorMenor objeto=new VectorMenor();
          int k, can,resul;

                //INGRESO
          System.out.print("Cuantos Elementos: ");
          can = leer.nextInt();

          int[] vec=new int[15];
          for(k=1;k<=can;k++){
                        vec[k] = (int)(Math.random()* 100);
                        System.out.print(vec[k] + ", ");
          }
          System.out.println("");
          resul = objeto.menor(vec, can);
                System.out.println("");
          System.out.println("El menor es: " + resul);
          System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el menor número de un vector de n elementos generado aleatoriamente, se usará clases externas.

TEMA 91: TRANSFORMAR NÚMERO DECIMAL A BINARIO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Binario {
        public void binarioDecimal(int num)  {
                int k, aux,pos;
                int[] vec =new int[15];
           pos = 1;
           while (num >= 2)
           {
                        aux = num % 2;
                        vec[pos] = aux;
                        pos = pos + 1;
                        num = num / 2;
           }
           vec[pos] = num;
           //SALIDA
           System.out.println ("\nNúmero Binario: ");
           for(k=pos;k>=1;k--)
                        System.out.print(vec[k]+" ");
        }
}
import java.util.Scanner;
public class TransformacionDecimalBinario_Clase {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        Binario objeto=new Binario();
                int num1;

                System.out.print("Digite un número : ");
      num1 = leer.nextInt();

      objeto.binarioDecimal(num1);
      System.out.println("");
                System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza la transformación de un número decimal ingresado por teclado en un número binario, para ello usaremos clases externas.

TEMA 92: OBTENER EL DÍGITO VERIFICADOR DE LA CÉDULA

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Cedula {
        public int DVCedula(int[] ced)  {
                int sumP, sumI, st, aux,i, digito;

                //SUMATORIA DE PARES
                sumP = 0 ; sumI = 0;
                for(i=2;i<=8;i+=2)
                        sumP = sumP + ced[i];

                //SUMATORIA DE IMPARES
                for (i=1;i<=9;i+=2)
                {
                        aux = ced[i]*2;
                        if (aux > 9)
                                aux = aux - 9;
                        sumI = sumI + aux;
                }

                //OBTENCIÓN DE DÍGITO
                st = sumP + sumI;
                digito = 10 - (st % 10);

                if (digito == 10)
                        digito = 0;
        return digito;
        }
}
import java.util.Scanner;
public class DigitoVerificadorCedula_Clase {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        Cedula objeto=new Cedula();
                int dv,i;
                int[] miCedu=new int[15];

                System.out.println("Digite los 9 primero dígitos de su cédula: ");
                for (i=1;i<=9;i++)
                {
                        System.out.print("Posición "+i+"==> ");
                        miCedu[i] = leer.nextInt();
                }
```

```
                dv = objeto.DVCedula(miCedu);
                System.out.println("El dígito verificador es: " + dv);
                System.out.println("");
                System.out.println("Gracias.");
    }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa realiza el cálculo del digito verificador de la cédula, se usara clases externas.

TEMA 93: ORDENAMIENTO DE UN ARREGLO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class VectorOrdenado {
        public void ordenar(int[] nuevo,int n){
                int i, k, aux ;
                for(i=1;i<=n;i++){
                        for(k=i;k<=n;k++){
                                if (nuevo[k] < nuevo[i]){
                                        aux = nuevo[k];
                                        nuevo[k] = nuevo[i];
                                        nuevo[i] = aux;
                                }
                        }
                }
        }
}
import java.util.Scanner;
public class OrdenamientoArreglo_Clase {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        VectorOrdenado objeto= new VectorOrdenado();
                int k, can;

                //INGRESO
                System.out.print("Cuantos elementos tendrá el Vector: ");
      can= leer.nextInt();

      int[] vec=new int[15];
      for(k=1;k<=can;k++){
                        vec[k] = (int)(Math.random() * 100);
                        System.out.print(vec[k] + ", ");
      }
      System.out.println("");

                // LLAMAMOS AL PROCEDIMIENTO
      objeto.ordenar(vec, can);
      System.out.println("\nArreglo Ordenado: ");
                for(k=1;k<=can;k++)
                System.out.print(vec[k] + ", ");
      System.out.println("");
      System.out.println("Gracias.");
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa ordena un vector de N elementos generado aleatoriamente, se usa clases externas.

TEMA 94: EJEMPLO DE FECHAS

1. Abrir un nuevo Proyecto
2. Código de la clase

```
import java.util.Date;
public class Fechas {

   public static void main(String[] args) {

        int h,m,s;
                Date d4=new Date();

                        h=d4.getHours();
                        m=d4.getMinutes();
                        s=d4.getSeconds();
                        System.out.println(h+" : "+m+" : "+s);

                        String cadena=d4.toLocaleString();
                        System.out.println(cadena);

                        Date d2=new Date(71,7,1);
                        System.out.println("Fecha segunda: "+d2);

                        Date d3=new Date("April 3 1993 3:24 PM");
                        System.out.println("Fecha tercera: "+ d3);

}
}
```

3. **GRABAR Y EJECUTAR**

FUNCIOMANIENTO

El programa nos indica el funcionamiento del metodo Date.

TEMA 95: ÁREA DEL TRIANGULO

1. Abrir un nuevo Proyecto
2. Código de la clase

```
public class Triangulo {
        float base,altura,area;
        public float area_triangulo()
        {
        area=base*altura/2;
        return area;
        }
}

import java.util.Scanner;
public class TrianguloBase {

   public static void main(String[] args) {
        Scanner leer=new Scanner(System.in);
        Triangulo r=new Triangulo();
           float resultado;
           System.out.print ("Ingrese la base: ");
           r.base=leer.nextFloat();
           System.out.print ("Ingrese la altura: ");
           r.altura=leer.nextFloat();
           resultado=r.area_triangulo();
           System.out.println("area del triangulo rectangulo = "+resultado);
   }
}
```

3. **GRABAR Y EJECUTAR**

FUNCIONAMIENTO

El programa calcula el area del triangulo, ingresando por teclado la base y altura; la clase realiza el calculo del area y nos retorna el mismo.

ESTRUCTURAS DE DATOS

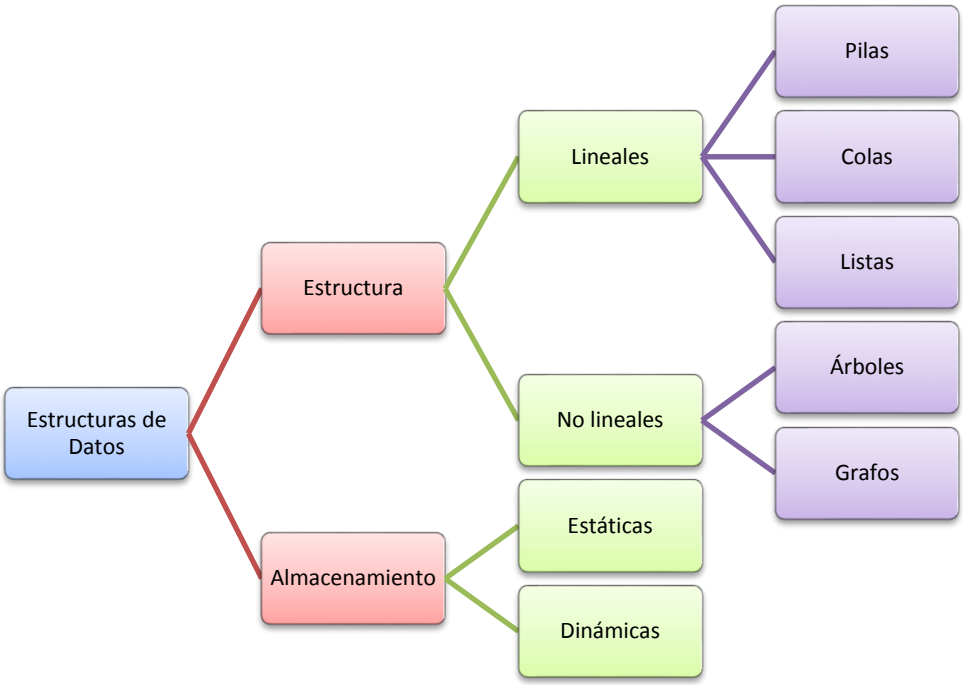
1. CONCEPTO Y CLASIFICACIÓN

Las estructuras de datos son un tipo especial de variable que permite el tratamiento y manipulación de un conjunto de datos del mismo tipo de una manera eficiente para optimizar el uso de memoria dentro del computador.

Las estructuras de datos se clasifican en función de dos criterios:

- Por su estructura
- Por la forma de almacenamiento

Éstas a su vez se sub clasifican de la siguiente manera:



Las estructuras de datos lineales almacenan los datos de manera secuencial, mientras que las no lineales organizan la información para ser accesible de manera aleatoria.

Las estructuras de datos Estáticas son aquellas que se implementan a través de arrays, predefinen el número de elementos que tendrá y por tanto reservan un espacio de memoria en su definición, el mismo que será fijo. Las estructuras de datos dinámicas van administrando el espacio que utilizan en memoria según sus necesidades y no tienen límite de almacenamiento de datos.

En base a las dos clasificaciones se pueden tener estructuras de datos lineales estáticas, lineales dinámicas, no lineales estáticas y no lineales dinámicas.

Para el presente apartado se trabajará con las estructuras de datos lineales estáticas implementadas a través de arrays.

1. OPERACIONES BÁSICAS EN ESTRUCTURAS DE DATOS LINEALES

Sobre las estructuras de datos lineales se pueden realizar distintas operaciones:

Recorrido: Procesa cada elemento de la estructura, en dependencia de su algoritmo de recorrido.
Búsqueda: encuentra un elemento y devuelve la posición en la que está almacenado.
Inserción: Agrega un elemento a la estructura de datos.
Borrado: Elimina un elemento.
Ordenación: Ordena los elementos de la estructura.

2. PILAS

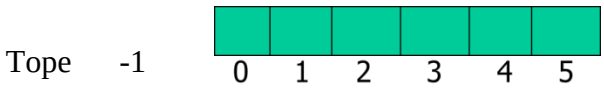
CONCEPTO

Es una estructura de datos lineal en donde los elementos se agregan y eliminan por el mismo extremo. Trabajan bajo el algoritmo de recorrido LIFO (Last In- First Out), es decir último en entrar, primero en salir.

Como ejemplos se pueden mencionar: una pila de platos, una pila de CDs, etc.

ESTRUCTURA

Implementada a través de arreglos es decir de manera estática:



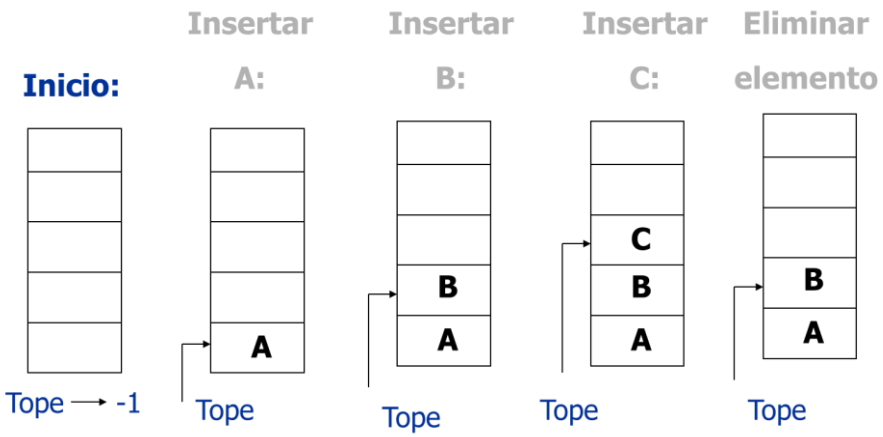
Se requiere contar con:

- Un arreglo unidimensional de n elementos.
- Un variable llamada tope que apunte siempre al último elemento de la pila y cuando la pila esté vacía apunte a -1.
- Controlar la posición máxima en la que se podrán almacenar valores (memoria estática).

OPERACIONES BÁSICAS

PUSH (insertar): Agrega un elemento a la pila a partir del último elemento (tope).
POP (extraer): Elimina el elemento de la pila que se encuentra en el tope.
VACIA: Verifica si la pila contiene o no elementos.
LLENA: Verifica si el último elemento de la pila ha llegado a la posición máxima.

En el siguiente gráfico se pueden observar las operaciones con pilas:



IMPLEMENTACIÓN DE PILAS

Para implementar pilas se estructurará una Clase externa que contenga los atributos y métodos que permitan trabajar con las distintas operaciones de esta estructura de datos y posteriormente permitan su uso en la implementación de ejercicios de aplicación.

Los atributos de la clase Pilas serán los elementos que la estructuran y que se explicaron anteriormente y los métodos permitirán implementar las operaciones propias de las pilas.

A continuación la clase Pilas:

```
public class Pilas
{
    //ESTRUCTURA DE LA PILA
    //DEFINIDA A TRAVÉS DE LOS ATRIBUTOS

    int pila[]=new int [10];
    int tope=-1;
    final int MAX=9;


    //METODOS PARA EL TRATAMIENTO DE PILAS
    //BASADOS EN LAS OPERACIONES BÁSICAS CON PILAS

    // ********** PILA VACIA **********//
```

```
//metodo con valor de retorno y sin argumentos

public boolean vacia()
{
	if(tope==-1)
	{
		return true;
	}
	else
	{
		return false;
	}
}


// *************** PILA LLENA *************************//
//metodo con valor de retorno y sin argumentos

public boolean llena()
{
	if(tope==MAX)
	{
		return true;
	}
	else
	{
		return false;
	}
}

// ******************* INSERTAR O PUSH *******************//
//metodo sin valor de retorno y con argumentos

public void push(int valor)
{
	//verificar si la pila esta llena
	if(llena())
	{
		System.out.println ("\n\nNo se pueden insertar elementos en la pila ....\n");
	}
	else
	{
		tope++;
		pila[tope]=valor;
	}
}

//*********************EXTRAER O POP ***********************//
//metodo con valor de retorno y sin argumentos

public int pop()
{
	int b=-1;

	if(vacia())
	{
```

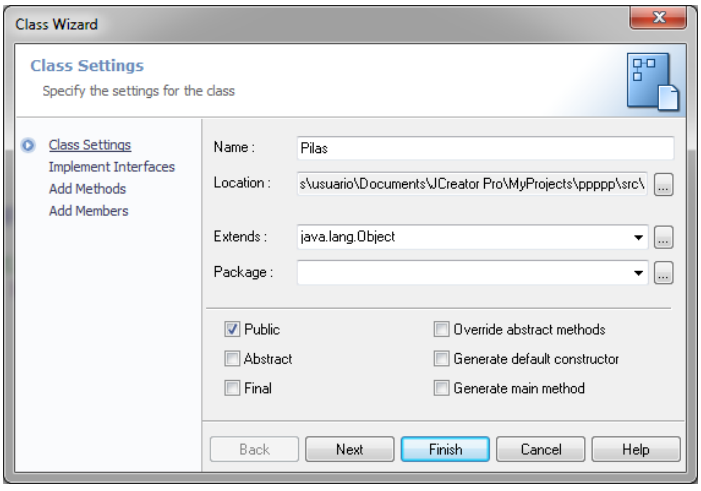
```
                System.out.println ("\n\nNo se pueden eliminar elementos ....\n");
        }
        else
        {
                b=pila[tope];
                pila[tope]=0;
                tope--;
        }
        return b;
    }

    //**************** METODO RECORRER ******************************
    //metodo sin valor de retorno y sin argumentos

    public void recorrer()
    {
        if(vacia())
        {
                System.out.println ("\n\nNo hay elementos para imprimir .....\n\n");
        }
        else
        {
                for(int i=tope;i>=0;i--)
                {
                        System.out.println (pila[i]);
                }
        }
    }
}
```

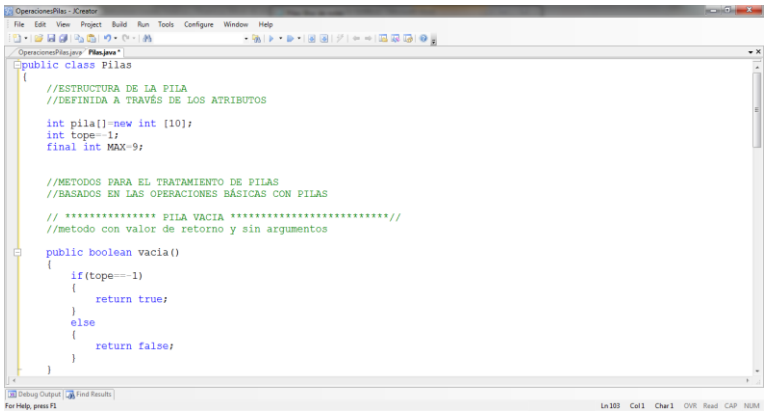
Para trabajar con la clase será necesario crear un nuevo proyecto en Jcreator como se ha realizado en todos los ejercicios, luego de lo cual se deberá incluir una clase de la siguiente manera:

Menú File / New / Class



Se coloca el nombre dela clase “Pilas”, se define la clase como pública para que sea accesible desde cualquier otra clase, finalmente presionamos Finish y allí se colocará el código de la clase.

Como se observa, se tiene la clase que contiene el método main y la clase Pilas que contiene la estructura de datos con sus atributos y métodos.



```
public class Pilas
{
    //ESTRUCTURA DE LA PILA
    //DEFINIDA A TRAVÉS DE LOS ATRIBUTOS

    int pila[]=new int [10];
    int tope=-1;
    final int MAX=9;


    //METODOS PARA EL TRATAMIENTO DE PILAS
    //BASADOS EN LAS OPERACIONES BÁSICAS CON PILAS

    // ************** PILA VACIA *************************//
    //metodo con valor de retorno y sin argumentos

    public boolean vacia()
    {
        if(tope==-1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

Para poder utilizar la clase Pilas, será necesario dentro de la clase principal crear una instancia de dicha clase, es decir crear un objeto que permita acceder a los elementos de la misma.

La creación del objeto se realizará de la siguiente manera:

```
Pilas pila1 = new Pilas();
```

El objeto pila1 servirá para poder utilizar los métodos de la clase, ya que una clase no es accesible directamente sino a través de un objeto.

A continuación el código de la clase principal con el cual se podrá probar la clase Pilas.

```
**
 * @(#)OperacionesPilas.java
 *
 * OperacionesPilas application
 *
 * @author
 * @version 1.00 2015/3/30
 */

import java.util.Scanner;
public class OperacionesPilas
{

   public static void main(String[] args)
   {
          Scanner leer = new Scanner(System.in);

          //declarar un objeto de la clase pilas para poder utilizar la estructura creada
```

```
//con los métodos respectivos

Pilas pila1 = new Pilas();
//se pueden definir n objetos de la clase Pilas
// ejemplo Pilas pila2 = new Pilas(); etcetera
int valor,op=0;


System.out.println ("\n\n\t\t\tOPERACIONES CON PILAS\n\n");

//VERIFICAR LA CONDICION PILA VACIA
System.out.println ("Verificar si la pila esta vacia ......");
if(pila1.vacia())
{
	System.out.println ("\n\nLa pila SI esta vacia .....\n\n");
}
else
{
	System.out.println ("\n\nLa pila NO esta vacia .....\n\n");
}

//VERIFICAR LA CONDICION PILA LLENA
System.out.println ("\nVerificar si la pila esta llena ......");
if(pila1.llena())
{
	System.out.println ("\n\nLa pila SI esta llena .....\n\n");
}
else
{
	System.out.println ("\n\nLa pila NO esta llena .....\n\n");
}

//INSERTAR ELEMENTOS EN LA PILA
System.out.println ("\n\nInsertar elementos a la pila ......\n");
do
{
	System.out.print ("\nIngrese el valor a insertar: ");
	valor=leer.nextInt();
	//insertando un valor a la pila
	pila1.push(valor);
	if(pila1.llena()==false)
	{
		System.out.print ("\nDesea agregar otro elemento a la pila <1/0>: ");
		op=leer.nextInt();
	}
}while(op==1 && pila1.llena()==false);


System.out.println ("\n\nLa pila ingresada es ......\n\n");
pila1.recorrer();

//EXTRAER ELEMENTOS DE LA PILA
System.out.println ("\n\nExtraer elementos de la pila ......\n");

do
{
```

```
                if(pila1.vacia()==false)
                {
                        System.out.println ("\nSe extrajo el elemento : "+pila1.pop());
                        System.out.println ("\n\nLa pila resultante es .....\n\n");
                        pila1.recorrer();
                }
                System.out.print ("\nDesea extraer otro elemento de la pila <1/0>: ");
                op=leer.nextInt();
        }while(op==1 && pila1.vacia()==false);

   }
}
```

3. COLAS

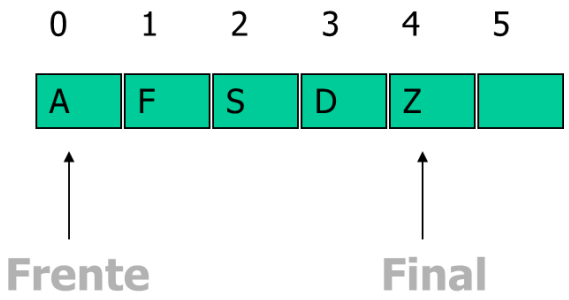
CONCEPTO

Es una estructura de datos lineal en donde los elementos se agregan y eliminan por diferentes extremos. Trabajan bajo el algoritmo de recorrido FIFO (First In- First Out), es decir primero en entrar, primero en salir.

Como ejemplos se pueden mencionar: una cola de un banco, una cola en una caja en un supermercado, etc.

ESTRUCTURA

Las colas implementadas con memoria estática requieren de un arreglo unidimensional (vector), además de dos variables que apunten al primero y al último elemento de la cola> frente y final o tope.



Se requiere contar con:

- Un arreglo unidimensional de n elementos.
- Un variable llamada final o tope que apunte siempre al último elemento de la cola y cuando la cola esté vacía apunte a -1.
- Una variable llamada frente que apunte siempre al primer elemento de la cola y cuando la cola esté vacía apunte a -1.
- Controlar la posición máxima en la que se podrán almacenar valores (memoria estática).

TIPOS DE COLAS

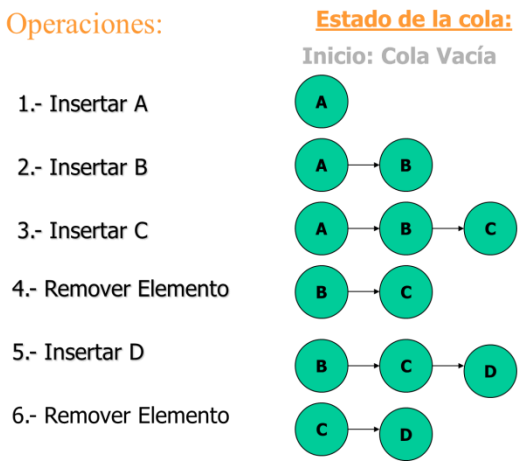
Tomando como base la forma en que trabaja la cola se pueden tener:

- **Colas con frente fijo**: los elementos se insertan por el tope y se extraen por el frente. Cuando se extrae un elemento, todos los que estén ubicados a continuación recorrerán una posición a la izquierda. El frente siempre se mantendrá en la posición cero del array.
- **Colas con frete móvil**: los elementos se insertan por el tope y se extraen por el frente. Cuando se extrae un elemento, el frente recorrerá una posición a la derecha. Si es que el frente no está en la posición cero pero el tope ha llegado a la máxima posición del arreglo, para poder insertar nuevamente un elemento se deberán recorrer todos los existentes una posición hacia la izquierda.
- **Colas Circulares:** los elementos se insertan por el tope y se extraen por el frente. Cuando se extrae un elemento, el frente recorrerá una posición a la derecha. Si es que el frente no está en la posición cero pero el tope ha llegado a la máxima posición del arreglo, para poder insertar nuevamente un elemento se deberá recorrer el tope hacia la posición cero formándose la cola circular.

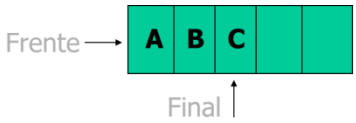
OPERACIONES BÁSICAS

Insertar: Almacena un elemento al final de la cola.
Eliminar: Extrae un elemento que se encuentra al frente.
Vacía: Verifica si la cola tiene o no elementos y devuelve un valor booleano.
Llena: Verifica si la cola tiene espacios disponibles para insertar nuevos elementos, retornando un valor booleano.

En el siguiente gráfico se pueden observar las operaciones en una cola frente fijo:



COLA FRENTE FIJO Y FRENTE MÓVIL

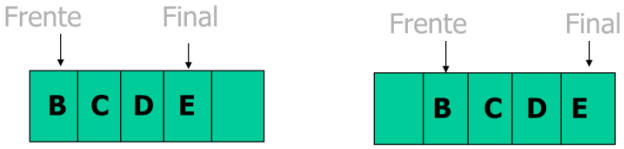


Al remover un elemento:

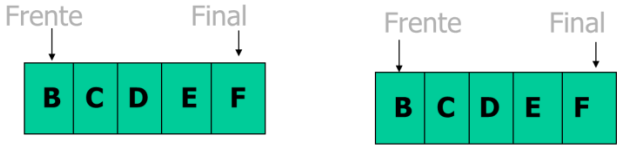
Frente fijo Frente movible

Insertar elemento D:

Insertar elemento E:

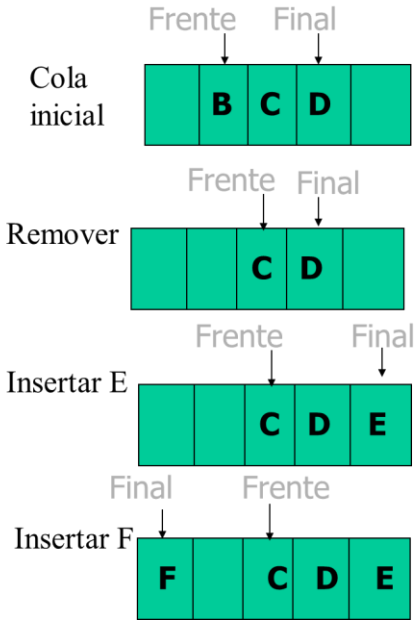


Insertar elemento F:



Insertar elemento G: Error: Cola llena!!!!

COLAS CIRCULARES



IMPLEMENTACIÓN DE COLAS

Para implementar colas se estructurará una Clase externa que contenga los atributos y métodos que permitan trabajar con las distintas operaciones de esta estructura de datos y posteriormente permitan su uso en la implementación de ejercicios de aplicación.

Los atributos de la clase Colas serán los elementos que la estructuran y que se explicaron anteriormente y los métodos permitirán implementar las operaciones propias de las pilas.

A continuación la clase Colas para cada uno de los tipos especificados:

COLAS FRENTE FIJO

Para trabajar se deberá crear un proyecto para las operaciones con colas, dentro de él se insertará una clase publica llamadas Colas en la que se deberá copia el código de la clase.

CLASE COLAS

```
public class Colas
{

    //ESTRUCTURA DE LA COLA

    int cola []=new int [10];
    int tope=-1;
```

```
int inicio =-1;
final int MAX =9;

//OPERACIONES CON COLAS CON FRENTE FIJO
//VERIFICAR SI LA COLA ESTA VACIA

public boolean vacia()
{
        if(tope==-1 && inicio ==-1)
        {
                return true;
        }
        else
        {
                return false;
        }
}


//VERIFICAR LA CONDICION COLA LLENA

public boolean llena()
{
        if(inicio ==0 && tope ==MAX)
        {
                return true;
        }
        else
        {
                return false;
        }
}

//INSERTAR ELEMENTO EN LA COLA

public void insertar (int valor)
{
        if(vacia())
        {
                inicio=0;
                tope++;
                cola[tope]=valor;
        }
        else
        {
                if(llena())
                {
                        System.out.println ("\n\nNo se pueden insertar elemementos ... cola
llena\n\n");
                }
                else
                {
                        tope++;
                        cola[tope]=valor;
                }
        }
```

```
    }
    //RECORRER LOS ELEMENTOS DE LA COLA ///// IMPRIMIR LA COLA

    public void recorrer()
    {
        if(vacia())
        {
            System.out.println ("\n\nNo hay elementos para imprimir\n\n");

        }
        else
        {
            for(int i=inicio;i<=tope;i++)
            {
                System.out.println (cola[i]);
            }
        }
    }


    //ELIMINAR ELEMENTOS DE LA COLA
    public int extraer ()
    {
        int borrado=-1,i;
        if(vacia())
        {
            System.out.println ("\n\nNo hay elementos que borrar ... cola vacia\n\n");
        }
        else
        {
            if(inicio==tope)
            {
                borrado=cola[inicio];
                inicio=-1;
                tope=-1;
            }
            else
            {
                borrado=cola[inicio];
                cola[inicio]=0; //borrado logico
                for(i=inicio;i<=tope-1;i++)
                {
                    cola[i]=cola[i+1];
                }
                tope--;
            }
        }
        return borrado;
    }
}
```

CLASE PRINCIPAL OPERACIONES CON COLAS FRENTE FIJO

Para poder utilizar la clase Colas, será necesario dentro de la clase principal crear una instancia de dicha clase, es decir crear un objeto que permita acceder a los elementos de la misma.

La creación del objeto se realizará de la siguiente manera:

Colas cola1 = new Colas();

El objeto cola1 servirá para poder utilizar los métodos de la clase.

A continuación el código de la clase principal con el cual se podrá probar la clase Colas.

```
/**
 * @(#)OperacionesColas.java
 *
 * OperacionesColas application
 *
 * @author
 * @version 1.00 2015/4/15
 */

import java.util.Scanner;
public class OperacionesColas
{

   public static void main(String[] args)
   {
     //objeto de lectra
     Scanner leer = new Scanner (System.in);

     //objeto de la clase colas
     Colas cola1 = new Colas();

     //variables
     int valor,op;

     System.out.println ("\n\n\t\t\tOPERACIONES CON COLAS DE FRENTE FIJO \n\n");

     //VERIFICAR LA CONDICION COLA VACIA

     if(cola1.vacia())
     {
               System.out.println ("\n\nLa cola SI esta vacia\n\n");
     }
     else
     {
               System.out.println ("\n\nLa cola NO esta vacia\n\n");
     }

     //VERIFICAR LA CONDICION COLA LLENA
```

```
        if(cola1.llena())
        {
                System.out.println ("\n\nLa cola SI esta llena\n\n");
        }
        else
        {
                System.out.println ("\n\nLa cola NO esta llena\n\n");
        }


        //insertar elementos
        System.out.println ("\n\nINSERTAR ELEMENTOS EN LA COLA\n\n");
        do
        {
                System.out.print ("\nValor a insertar: ");
                valor=leer.nextInt();
                cola1.insertar(valor);
                System.out.print ("\nDesea insertar otro valor <1/0>: ");
                op=leer.nextInt();
        }while(op==1 && cola1.llena()==false);

        //VISUALIZAR LOS ELEMENTOS DE LA COLA
        System.out.println ("\n\nLos elementos de la cola son .....\n\n");
        cola1.recorrer();

        //ELIMINAR ELEMENTOS DEL LA COLA

        System.out.println ("\n\nELIMINAR ELEMENTOS DE LA COLA\n\n");
        do
        {
                System.out.println ("\nSe ha eliminado el valor: "+cola1.extraer());
                System.out.println ("\nLos elementos que quedan son .... ");
                cola1.recorrer();
                System.out.print ("\n\nDesea extraer otro elemento <1/0>: ");
                op=leer.nextInt();
        }while(op==1 && cola1.vacia()==false);

    }
}
```

COLAS FRENTE MÓVIL

Se procederá de la misma manera que en las colas frente fijo, es decir creando la clase principal y la clase externa para estructurar la clase colas frente móvil.

CLASE COLAS FRENTE MÓVIL

```
public class ColasFrenteMovil
{
    //ATRIBUTOS

    int inicio=-1;
```

```
int tope=-1;
int cola[]=new int[10];
final int MAX =9;


//OPERACIONES

//CONTROLAR COLA VACIA

public boolean vacia()
{
        if(inicio==-1 && tope==-1)
        {
                return true;
        }
        else
        {
                return false;
        }
}

//CONTROL COLA LENA

public boolean llena()
{
        if(inicio==0 && tope==MAX)
        {
                return  true;
        }
        else
        {
                return false;
        }
}

//INSERTAR ELEMENTOS EN LA COLA

public void insertar (int valor)
{
        if(vacia())
        {
                inicio=0;
                tope=0;
                cola[tope]=valor;
        }
        else
        {
                if(llena())
                {
                        System.out.println ("\n\nNo se pueden insertar elementos ... cola
llena\n\n");
                }
                else
                {
                        if(inicio!=0 && tope==MAX)
                        {
```

```
                                        for(int i=inicio-1;i<=tope-1;i++)
                                        {
                                                cola[i]=cola[i+1];
                                        }
                                        inicio--;
                                        cola[tope]=valor;
                                }
                                else
                                {
                                        tope++;
                                        cola[tope]=valor;
                                }
                        }
                }
        }

        //EXTRAER ELEMENTOS DE LA COLA

        public int extraer()
        {
                int borrado=-1;
                if(vacia())
                {
                        System.out.println ("\n\nNo se pueden extraer elementos ... cola vacia\n\n");
                }
                else
                {
                        if(inicio==tope)
                        {
                                borrado=cola[inicio];
                                inicio=-1;
                                tope=-1;
                        }
                        else
                        {
                                borrado=cola[inicio];
                                cola[inicio]=0;
                                inicio ++;
                        }
                }
                return borrado;
        }

        //IMPRIMIR LOS ELMENETOS DE LA COLA
        public void recorrer()
        {
                if(vacia())
                {
                        System.out.println ("\n\nNo hay elementos para imprimir....\\n\n");
                }
                else
                {
                        for(int i=inicio;i<=tope;i++)
                        {
                                System.out.println ("["+i+"]: "+cola[i]);
                        }
```

```
                }
        }
}
```

CLASE PRINCIPAL OPERACIONES COLAS FRENTE MOVIL

```
/**
 * @(#)OperacionesColasFrenteMovil.java
 *
 * OperacionesColasFrenteMovil application
 *
 * @author
 * @version 1.00 2015/4/20
 */

import java.util.Scanner;
public class OperacionesColasFrenteMovil
{


   public static void main(String[] args)
   {
     Scanner leer=new Scanner(System.in);
     int op,op2;

     //OBJETO DE A CLASE COLAS FRENTE MOVIL
     ColasFrenteMovil cola1=new ColasFrenteMovil();

     do
     {
              System.out.println ("\n\n\t\t\tOPERACIONES CON COLAS CON FRENTE MOVIL\n\n");
              System.out.println ("1. Cola Vacia\n");
              System.out.println ("2. Cola llena\n");
              System.out.println ("3. Insertar\n");
              System.out.println ("4. Extraer\n");
              System.out.println ("5. Salir\n");
              System.out.print ("\nIngrese una opcion: ");
              op=leer.nextInt();

              switch(op)
              {
                       case 1:
                       {
                                System.out.println ("\n\nCONTROL COLA VACIA\n\n");
                                if(cola1.vacia())
                                {
                                         System.out.println ("\n\nLa cola esta vacia\n\n");
                                }
                                else
                                {
                                         System.out.println ("\n\nLa cola no esta vacia\n\n");
                                }
                                break;
                       }
```

```
case 2:
{
        System.out.println ("\n\nCONTROL COLA LLENA\n\n");
        if(cola1.llena())
        {
                System.out.println ("\n\nLa cola esta llena\n\n");
        }
        else
        {
                System.out.println ("\n\nLa cola no esta llena\n\n");
        }
        break;
}

case 3:
{
        int valor,op3=1;
        System.out.println ("\n\nINSERTAR ELEMENTOS EN LA COLA\n\n");
        while(cola1.llena()==false && op3==1)
        {
                System.out.print ("Ingrese el valor a insertar: ");
                valor=leer.nextInt();
                cola1.insertar(valor);
                System.out.print ("\nDesea insertar otro valor <1/0>: ");
                op3=leer.nextInt();
        }
        System.out.println ("\n\nLa cola insertada es .....\n\n");
        cola1.recorrer();
        break;
}

case 4:
{
        int op3=1;
        System.out.println ("\n\nEXTRAER ELEMENTOS DE LA COLA\n\n");
        while(cola1.vacia()==false && op3==1)
        {
                System.out.println ("El elemento borrado es: "+cola1.extraer());
                System.out.println ("\n\nLa cola resultante es ...\n\n");
                cola1.recorrer();
                System.out.print ("\n\nDesea eliminar otro elemento <1/0>: ");
                op3=leer.nextInt();
        }
        break;
}

case 5:
{
        System.out.println ("\n\nGracias por usar el programa .....\n\n");
        break;
}

default:
{
        System.out.println ("\n\nOpcion incorrecta\n\n");
        break;
```

```
                    }

                }

                System.out.print ("\n\nDESEA CONTINUAR EN EL MENU. . . . : ");
                op2=leer.nextInt();
        }while(op2==1);
    }
}
```

COLAS CIRCULARES

Se procederá de la misma forma que en os casos anteriores

CLASE COLAS CIRCULARES

```
class ColasCirculares
{
    //atributos
    int tope=-1;
    int frente=-1;
    final int MAX=9;
    int cola[]=new int[10];
    //METODOS

    //cola vacia
    public boolean vacia()
    {
        if(tope==-1 && frente==-1)
            return true;
        else
            return false;
    }


    //COLA LLENA
    public boolean llena()
    {
        if(frente==0 && tope==MAX)
            return true;
        else
            if (frente==tope+1)
                return true;
            else
                return false;
    }


    //METODO INSERTAR
    public void insertar(int dato)
    {
        boolean x,y;
        x=vacia();
        y=llena();
```

```
            if(y==true)
            {
                System.out.println("NO SE PUEDEN INSERTAR ELEMENTOS");
            }
            else
            {
                if(x==true)
                {
                    frente=0;
                    tope=0;
                    cola[0]=dato;
                }

                else
                {
                    if(tope<MAX && tope>=frente)
                    {
                        tope++;
                        cola[tope]=dato;
                    }
                    else
                    {
                        if(tope==MAX&&frente!=0)
                        {
                            tope=0;
                            cola[tope]=dato;
                        }
                        else
                        {
                            if(tope<MAX&&tope<frente)
                            {
                                tope++;
                                cola[tope]=dato;
                            }
                        }
                    }
                }
    }
}


    //METODO EXTRAER RETORNANDO EL VALOR QUE SE EXTRAJO DE LA COLA

    public int extraer ()
    {
        boolean x;
        x=vacia();
        int borrado;

        //comprobar si la cola esta vacia
        if(x==true)
        {
            System.out.println("No se pueden extraer elementos pues la cola esta vacia");
            return -1;
        }
        else
```

```
        {
                //VERIFICAR SI EL FRENTE ES MENOR QUE EL TOPE Y QUE EL VALOR
MAXIMO
                if((frente>=0)&& (frente<tope ||frente<MAX))
                {
                        borrado=cola[frente];
                        cola[frente]=-1;
                        frente++;
                        return borrado;
                }
                else
                {
                        //VERIFICAR SI EL FRENTE ESTA UBICADO EN LA POSICION
MAXIMA DE LA COLA
                        if(frente==MAX)
                        {
                                borrado=cola[frente];
                                cola[frente]=-1;
                                frente=0;
                                return borrado;
                        }
                        else
                        {
                                //VERIFICAR SI EN LA COLA EXISTE UN SOLO
ELEMENTO
                                if(frente==tope)
                                {
                                        borrado=cola[frente];
                                        cola[frente]=-1;
                                        frente=-1;
                                        tope=-1;
                                        return borrado;

                                }
                                else
                                        return -1;
                        }
                }
        }
    }


    //METODO IMPRIMIR CONSIDERANDO QUE EL TOPE SE MAYOR QUE EL FRENTE O
QUE SEA MENOR QUE EL FRENTE

    public void imprimir()
    {
        boolean x;

        x=vacia();
        int i,j;

        //VERIFICAR SI LA COLA ESTA VACIA

        if(x==true)
```

```
            {
                System.out.println("La cola no tiene elementos");
            }
            else
            {
                //VERIFICAR QUE EL FRENTE SEA MENOS QUE EL TOPE

                if(frente >=0 && frente <=tope)
                {
                    for(i=frente;i<=tope;i++)
                    {
                        System.out.println("Elemento["+i+"]: "+cola[i]);
                    }
                }
                else
                {
                    //VERIFICAR SI EL TOPE ESTA ANTES DEL FRENTE ES DECIR SI
LA COLA GIRO DE MANERA CIRCULAR
                    if(frente <=MAX &&tope<frente)
                    {
                        //ESTE FOR IMPRIME LOS VALORES DESDE EL FRENTE
HASTA EL VALOR MAXIMO
                        for(i=frente;i<=MAX;i++)
                        {
                            System.out.println("Elemento["+i+"]: "+cola[i]);
                        }

                        //ESTE FOR IMPRIME LOS VALORES DESDE 0 HASTA
TOPE
                        for(j=0;j<=tope;j++)
                        {
                            System.out.println("Elemento["+j+"]: "+cola[j]);
                        }
                    }
                }
            }
        }
}
```

CLASE PRINCIPAL OPERACIONES CON COLAS CIRCULARES

```
**
 * @(#)cola_circular.java
 *
 * cola_circular application
 *
 * @author
 * @version 1.00 2012/4/3
 */

 import java.util.Scanner;
```

```
public class OperacionesColasCirculares
{

   public static void main(String[] args)
   {

            Scanner leer=new Scanner(System.in);
            int dato,op;

            //objeto para manipular la clase Colas
            ColasCirculares cola1=new ColasCirculares();

            //CRER MENU PARA TRABAJO REPETITIVO CON COLAS CIRCULARES

            do //ESTE CICLO CONTROLARA QUE SE REPITA EL MENU
            {

                  //CREAR EL MENU DE COLAS
                  System.out.println("\t\t\tMENU PARA EL TRABAJO CON COLAS
CIRCULARES");
                  System.out.println("\t\t\t**** **** ** ******* *** ***** **********\n\n");
                  System.out.println("1. COLA VACIA");
                  System.out.println("2. COLA LLENA");
                  System.out.println("3. INSERTAR UN ELEMENTO");
                  System.out.println("4. EXTRAER UN ELEMENTO");
                  System.out.println("5. SALIR\t");

                  //LEER LA OPCION PARA EL SWITCH

                  System.out.println("Ingrese la opcion: ");
                  op=leer.nextInt();

                  //VALIDAR EL SWITCH

                  switch(op)
                  {
                        case 1:
                        {
                              System.out.println("\t\tCOLA VACIA\n");
                              boolean x=cola1.vacia();
                              //VERIFICAR QUE LA COLA ESTA VACIA
                              if(x==true)
                                    System.out.println("LA COLA ESTA
VACIA.....\n\n");
                              break;
                        }
                        case 2:
                        {
                              System.out.println("\t\tCOLA LLENA\n");
                              //VERIFICAR SI LA COLA ESTA LLENA
                              boolean y=cola1.llena();
                              if(y==true)
                                    System.out.println("LA COLA ESTA
LLENA.....\n\n");
                              break;
```

```
                                }
                                case 3:
                                {
                                        System.out.println("\t\tINSERTAR ELEMENTO\n");
                                        System.out.println("Ingrese el elemento que desea insertar en la
cola: ");
                                dato=leer.nextInt();
                                cola1.insertar(dato);
                                System.out.println("\nLa cola resultante es: ");
                                cola1.imprimir();
                                break;
                                }
                                case 4:
                                {
                                        System.out.println("\t\tEXTRAER ELEMENTO\n");

                                        System.out.println("El elemento que se borro de la cola es:
"+cola1.extraer()+"\n");
                                System.out.println("\nLa cola resultante es: ");
                                cola1.imprimir();
                                break;

                                }
                                case 5:
                                {
                                        System.out.println("\t\tSALIR .....\n");
                                        break;

                                }
                                default:
                                {
                                        System.out.println("\t\tOPCION INCORRECTA !!!! .....\n");

                                }
                        }

                }while(op>=1 && op<=4);
        }
}
```

4. LISTAS

CONCEPTO

Es una colección lineal de datos almacenados en posiciones consecutivas de memoria y que se caracterizan por ser estructuras flexibles ya que no responden a ningún algoritmo de recorrido.

ESTRUCTURA

Para su implementación utilizando arreglos se requiere:

- Un arreglo unidimensional (vector)
- Un elemento llamado **tope** que apunta al último elemento de la lista.
- La referencia de la posición máxima de la lista (memoria estática).

OPERACIONES

Las listas al ser estructuras de datos tan flexibles permiten el desarrollo de múltiples operaciones:

- Verificar si la lista está vacía o si está llena
- Crear la lista, agregando elementos a partir del tope.
- Recorrer la lista desde el primero hasta el último elemento.
- Buscar un elemento en la lista, se puede hacer a partir de un valor en cuyo caso retornaría la posición en la que se encuentra el elemento, ó a partir de una posición existente en cuyo caso retornaría el valor almacenado.
- Insertar un elemento a partir de una posición en donde los elementos contiguos recorrerían hacia a derecha ó a partir de un valor en donde ocurriría lo mismo.
- Ordenar los elementos de la lista ya sea de forma ascendente o descendente.
- Extraer un elemento de la lista, para lo cual los elementos contiguos al eliminado deberían recorrer hacia la izquierda.

IMPLEMENTACIÓN

Para implementar las listas se trabajará con una clase externa con los atributos y métodos que permitan trabajar con ella y una clase principal en la que se creará la instancia de la clase Listas y partir de dicha instancia se acceda a las operaciones correspondientes.

CLASE LISTAS

```
public class Listas
{

    //ATRIBUTOS DE LA CLASE
    String lista[]=new String[10];
    int tope=-1;
    final int MAX=9;


    //metodos para implementar operaciones con listas

    public boolean lista_vacia()
    {
            if(tope==-1)
            {
                    return true;
            }
```

```
		else
		{
			return false;
		}
	}

	public boolean lista_llena()
	{
		if(tope==MAX)
		{
			return true;
		}
		else
		{
			return false;
		}
	}

	//metodo para crear en una lista de manera secuencial

	public void crear_lista(String valor)
	{
		if(lista_vacia())
		{
			tope=0;
			lista[tope]=valor;
		}
		else
		{
			if(lista_llena())
			{
				System.out.println ("\nLa lista esta llena ... no se pueden insertar
elementos\n");
			}
			else
			{
				if(tope<MAX)
				{
					tope++;
					lista[tope]=valor;
				}
			}
		}
	}

	//IMPRIMIR LOS ELEMENTOS DE LA LISTA

	public void imprimir()
	{
		if(lista_vacia())
		{
			System.out.println ("\nNo existen elementos en la lista\n");
		}
		else
		{
			System.out.println ("\n\n\t\t\tElementos de la lista \n\n");
```

```
                for(int i=0;i<=tope;i++)
                {
                        System.out.println ("Elemento "+i+": "+lista[i]);
                }
        }
    }

    //INSERTAR UN VALOR A PARTIR DE UN POSICION DETREMINADA

    public void insertar_posicion(int pos, String valor)
    {
        if(lista_vacia()==true && pos==0)
        {
                tope=0;
                lista[tope]=valor;
        }
        else
        {
                if(lista_llena())
                {
                        System.out.println ("\n\nNo se puede insertar el elemento .... Lista
llena\n\n");
                }
                else
                {
                        if(pos>=0 && pos<=tope)
                        {
                                for(int i=tope;i>=pos;i--)
                                {
                                        lista[i+1]=lista[i];
                                }
                                lista[pos]=valor;
                                tope++;
                        }
                        else
                        {
                                System.out.println ("\n\nLa posicion no es la correcta ...\n\n");

                        }
                }
        }
    }

    //METODOS PARA REALIZAR BUSQUEDAS

    //ENVIO EL VALOR A BUSCAR Y RETORNA LA POSICION EN LA QUE LO ENCONTRO

    public int buscarPorValor(String valor)
    {
        int i,pos=-1,b=0;
        for(i=0;i<=tope && b==0;i++)
        {
                if(lista[i].equals(valor))
                {
                        pos=i;
                        b=1;
```

```
                }
        }
        return pos;
}

//BUSCAR A PARTIR DE UNA POSICION Y RETORNAR EL VALOR

public String buscarPorPos(int pos)
{
        String valor;
        int i;
        if(pos>=0 && pos<=tope)
        {
                return lista[pos];
        }
        else
        {
                return "";
        }
}

//BUSCAR UN ELEMENTO ESPECIFICO EN UNA POSICION DETERMINADA

public boolean buscarElemPos(String valor, int pos)
{
        if(pos>=0 && pos<=tope)
        {
                if(lista[pos].equals(valor))
                {
                        return true;
                }
                else
                {
                        return false;
                }
        }
        else
        {
                return false;
        }
}

//BUSCAR PRIMERA OCURRENCIA DE UN ELEMENTO EN LA LISTA

public int primera_ocurrencia(String valor)
{
        return buscarPorValor(valor);
}



//CONTAR EL NUMERO DE OCURRENCIAS DE UN ELEMENTO EN LA LISTA

public int contar_elementos(String valor)
{
        int i,cont=0;
```

```
		if(lista_vacia())
		{
			System.out.println ("\n\nLa lista esta vacia .... no se pueden contar los
elementos\n");
		}
		else
		{
			if(buscarPorValor(valor)!=-1)
			{
				for(i=0;i<=tope;i++)
				{
					if(lista[i].equals(valor))
					{
						cont++;
					}
				}
			}
		}
		return cont;

	}


	//BUSCAR ULTIMA OCURRENCIA DE UN ELEMENTO EN LA LISTA

	public int ultima_ocurrencia(String valor)
	{
		int i,b=0,pos=-1;

		if(lista_vacia())
		{
			System.out.println ("\n\nNo se puede buscar el elemento.... lista vacia\n");

		}
		else
		{
			if(contar_elementos(valor)>1)
			{
				for(i=tope;(i>=0 && b==0);i--)
				{
					if(lista[i].equals(valor))
					{
						pos=i;
						b=1;
					}
				}

			}
			else
			{
				pos=buscarPorValor(valor);
			}

		}
		return pos;
```

```
}

//INSERTAR ELEMENTO A PARTIR DE UN VALOR DETERMINADO

public void insertar_valor(String valorb, String valori)
{

	int pos;

	if(lista_llena())
	{
		System.out.println ("\n\nLa lista esta llena ... no se pueden insertar elementos\n");
	}
	else
	{
		pos=buscarPorValor(valorb);
		if(pos!=-1)
		{
			for(int i=tope;i>=pos;i--)
			{
				lista[i+1]=lista[i];
			}
			lista[pos+1]=valori;
			tope++;
		}
		System.out.println ("\n\nLista resultante.....\n\n");
		imprimir();
	}
}

//ORDENAR LISTA ASCENDENTE Y DESCENDENTEMENTE

public void ordenar_lista(int tipoorden)
{
	int i,j;
	String aux;

	if(tipoorden==1)
	{
		for(i=0;i<=tope-1;i++)
		{
			for(j=i+1;j<=tope;j++)
			{
				int x=lista[j].compareTo(lista[i]);
				if(x<0)
				{
					aux=lista[j];
					lista[j]=lista[i];
					lista[i]=aux;
				}
			}
		}
	}
	else
	{
		for(i=0;i<=tope-1;i++)
```

```
                {
                        for(j=i+1;j<=tope;j++)
                        {
                                int x=lista[j].compareTo(lista[i]);
                                if(x>0)
                                {
                                        aux=lista[j];
                                        lista[j]=lista[i];
                                        lista[i]=aux;
                                }
                        }
                }

        }

    }


    //EDITAR UN ELEMENTO DE LA LISTA

    public void editar(String valor, String nuevo)
    {
        int pos=buscarPorValor(valor);
        if(pos!=-1)
        {
                lista[pos]=nuevo;
                System.out.println ("\n\nLista resultante ....\n");
                imprimir();
        }
        else
        {
                System.out.println ("\n\nEl valor solicitado no existe ....\n\n");
        }
    }
}
```

CLASE ADICIONAL

Para trabajar de mejor manera se creará una clase adicional tal como se procedió con la clase listas, la misma que contendrá los menúes de acceso en el programa principal.

```
public class Menues
{
    //SUBMENUS DE OPCIONES

    public void submenu_buscar()
    {
        System.out.println ("\n\n\t\t\tBUSQUEDAS\n\n");
        System.out.println ("1. Buscar por valor\n");
        System.out.println ("2. Buscar por posicion\n");
        System.out.println ("3. Buscar elemento en una posicion determinada\n");
        System.out.println ("4. Buscar primera ocurrencia de un elemento\n");
        System.out.println ("5. Buscar ultima ocurrencia de un elemento\n");
        System.out.println ("6. Contar el numero de ocurrencias de un elemento\n");
        System.out.println ("7. Regresar al menu principal\n");
```

```
    }

    public void submenu_ordenar()
    {
            System.out.println ("\n\n\t\t\tORDENAMIENTO\n\n");
            System.out.println ("1. Ascendente\n");
            System.out.println ("2. Descendente\n");
            System.out.println ("3. Regresar al menu principal\n");

    }
}
```

CLASE PRINCIPAL OPERACIONES CON LISTAS

```
/**
 * @(#)OperacionesListas.java
 *
 * OperacionesListas application
 *
 * @author
 * @version 1.00 2013/11/61
 *
 */

import java.util.Scanner;

public class OperacionesListas
{

   public static void main(String[] args)
   {
      //OBJETO PARA LECTURA
      Scanner leer = new Scanner(System.in);

      //OBJETO DE LA CLASE LISTAS
      Listas lista1=new Listas();

      //OBJETO PARA MANEJAR LOS MENUES
      Menues pantallas =new Menues();

      //VARIABLES AUXILIARES
      String valor;
      int pos,op1,op2,i,op3=0,op4=0,op5=0,op6=0,op7=0;

      System.out.println ("\n\n\t\t\tOPERACIONES CON LISTAS\n\n");

      do
      {
            System.out.println ("\n\n\t\t\tMENU DE OPCIONES\n\n");
            System.out.println ("1. Crear Lista\n");
            System.out.println ("2. Imprimir Lista\n");
            System.out.println ("3. Insertar elemento en una posicion\n");
            System.out.println ("4. Insertar elemento a partir de un valor\n");
            System.out.println ("5. Buscar elemento\n");
            System.out.println ("6. Ordenar la lista\n");
            System.out.println ("7. Editar un elemento de la lista\n");
            System.out.println ("8. Salir\n");
            System.out.print ("\n\nSeleccione la opcion: ");
```

```
op1=leer.nextInt();

switch(op1)
{
	case 1:
	{
		System.out.println ("\n\n\t\t\tCREACION DE LA LISTA DE NOMBRES\n\n");
		do
		{

			System.out.print ("\nIngrese su nombre: ");
			valor=leer.next();
			lista1.crear_lista(valor);
			System.out.print ("\nDesea agregar otro nombre <1/0> : ");
			op2=leer.nextInt();
		}while(op2==1);
		break;
	}
	case 2:
	{
		System.out.println ("\n\n\t\t\tIMPRIMIR LA LISTA DE NOMBRES\n\n");
		lista1.imprimir();
		break;
	}
	case 3:
	{
		System.out.println ("\n\n\t\tINSERTAR UN ELEMENTO EN UNA POSICION
DETERMINADA\n\n");
		System.out.print ("Ingrese la posicion: ");
		pos=leer.nextInt();
		System.out.print ("Ingrese el Nombre a insertar: ");
		valor=leer.next();
		lista1.insertar_posicion(pos,valor);
		break;
	}
	case 4:
	{
		String valori;
		System.out.println ("\n\n\t\tINSERTAR UN ELEMENTO A PARTIR DE UN
VALOR DETERMINADO\n\n");
		System.out.print ("Valor a partir del cual va a insertar : ");
		valor=leer.next();
		System.out.print ("Ingrese el Nombre a insertar: ");
		valori=leer.next();
		lista1.insertar_valor(valor,valori);
		break;
	}
	case 5:
	{
		do
		{
			pantallas.submenu_buscar();
			System.out.print ("\nElija la opcion: ");
			op4=leer.nextInt();
			switch(op4)
			{
				case 1:
				{
					System.out.println ("\n\n\t\t\tBusqueda por
Valor\n\n");

					System.out.print ("Ingrese el valor a buscar: ");
					valor=leer.next();
```

```
                                        if(lista1.buscarPorValor(valor)==-1)
                                        {
                                                System.out.println ("\nEse valor no existe
en la lista\n");
                                        }
                                        else
                                        {
                                                System.out.println ("\nEl valor: "+valor+"
esta en la posicion "+lista1.buscarPorValor(valor));
                                        }
                                        break;
                                }
                                case 2:
                                {
                                        System.out.println ("\n\n\t\t\tBusqueda de un valor a
partir de una posicion\n\n");
                                        System.out.print ("Ingrese la posicion: ");
                                        pos=leer.nextInt();
                                        if(lista1.buscarPorPos(pos).equals(""))
                                        {
                                                System.out.println ("\n\nEsa posicion no
existe en la lista\n\n");
                                        }
                                        else
                                        {
                                                System.out.println ("\n\nLa posicion
"+pos+" contiene "+lista1.buscarPorPos(pos));
                                        }
                                        break;

                                }
                                case 3:
                                {
                                        System.out.println ("\n\n\t\t\tBusqueda de un valor en
una posicion determinada\n\n");
                                        System.out.print ("Ingrese la posicion: ");
                                        pos=leer.nextInt();
                                        System.out.print ("Ingrese el valor a buscar: ");
                                        valor=leer.next();
                                        if(lista1.buscarElemPos(valor,pos))
                                        {
                                                System.out.println ("\nEl valor "+valor+" si
esta en la posicion "+pos);
                                        }
                                        else
                                        {
                                                System.out.println ("El valor no se
encontro en esa posicion\n");
                                        }
                                        break;

                                }
                                case 4:
                                {
                                        System.out.println ("\n\n\t\t\tBuscar la primea
ocurrencia de un valor\n\n");
                                        System.out.print ("Ingrese el valor a buscar: ");
                                        valor=leer.next();
                                        System.out.println ("\n\nLa primea posicion en la que
aparece "+valor+" es "+lista1.primera_ocurrencia(valor));
                                        break;
```

```
                                        }
                                        case 5:
                                        {
                                                System.out.println ("\n\n\t\t\tBuscar la ultima
ocurrencia de un valor\n\n");
                                                System.out.print ("Ingrese el valor a buscar: ");
                                                valor=leer.next();
                                                System.out.println ("\n\nLa ultima posicion en la que
aparece "+valor+" es "+lista1.ultima_ocurrencia(valor));
                                                break;

                                        }
                                        case 6:
                                        {
                                                System.out.println ("\n\n\t\t\tContar el número de
ocurrencia de un elemento\n\n");
                                                System.out.print ("Ingrese el valor a buscar: ");
                                                valor=leer.next();
                                                System.out.println ("\n\nEl elemento aparece
"+lista1.contar_elementos(valor)+" veces\n");
                                                break;

                                        }
                                        case 7:
                                        {
                                                op5=0;

                                                break;

                                        }
                                        default:
                                        {
                                                System.out.println ("\n\nOpcion incorrecta\n\n");
                                        }
                                }
                                if(op4!=7)
                                {
                                        System.out.print ("\nDesea continuar en el menu de busquedas:
");
                                        op5=leer.nextInt();
                                }
                                else
                                {
                                        op3=1;
                                }
                        }while(op5==1);
                        break;
                }

                case 6:
                {
                        do
                        {
                                pantallas.submenu_ordenar();
                                System.out.print ("\nIngrese la opcion: ");
                                op6=leer.nextInt();
                                switch(op6)
                                {
                                        case 1:
                                        {
                                                System.out.println ("\n\n\t\t\tORDENAMIENTO
ASCENDENTE\n");
```

```
                                        lista1.ordenar_lista(1);
                                        System.out.println ("\n\n La lista ordenada es ....\n");
                                        lista1.imprimir();
                                        break;
                                }
                                case 2:
                                {
                                        System.out.println ("\n\n\t\t\tORDENAMIENTO
DESCENDENTE\n");
                                        lista1.ordenar_lista(2);
                                        System.out.println ("\n\n La lista ordenada es ....\n");
                                        lista1.imprimir();
                                        break;
                                }
                                case 3:
                                {
                                        op5=0;
                                        break;
                                }
                                default:
                                {
                                        System.out.println ("\n\nOpcion incorrecta .....\n\n");
                                        break;
                                }
                        }
                        if(op6!=3)
                        {
                                System.out.print ("\n\nDesea seguir ordenando la lista <1/0>:
");
                                op5=leer.nextInt();
                        }
                }while(op5==1);
                break;

        }

        case 7:
        {
                String nuevo;
                do
                {
                        System.out.println ("\n\n\t\t\tEDITAR UN ELEMENTO DE LA
LISTA\n\n");
                        System.out.println ("\nLista original ....\n");
                        lista1.imprimir();
                        System.out.print ("\n\nQue valor desea modificar: ");
                        valor=leer.next();
                        System.out.print ("\nIngrese el nuevo valor: ");

                        nuevo=leer.next();
                        lista1.editar(valor,nuevo);
                        System.out.print ("\n\nDesea modificar otro valor: ");
                        op7=leer.nextInt();
                }while(op7==1);
                break;
        }
        case 8:
        {
                System.out.println ("\n\nGracias por usar el programa ....\n\n");
                op3=0;
                break;
        }
```

```
                        default:
                        {
                                System.out.println ("\n\nOpcion Incorrecta!! .....\n\n");
                                break;
                        }
                }

                if(op1!=8)
                {
                        if(op4!=7 || op5!=1)
                        {
                                System.out.print ("\n\nDesea regresar al Menu de opciones <1/0> : ");
                                op3=leer.nextInt();
                        }
                }

        }while(op3==1);

    }
}
```

5. EJERCICIOS CON ESTRUCTURAS DE DATOS

Para los ejercicios con las distintas estructuras de datos se trabajará con las clases definidas anteriormente las mismas que deberán ser usadas según corresponda como clase externas.

5.1.EJERCICIOS CON PILAS

PILAS SOMBRERO

```
/**
 * @(#)PilasSombrero.java
 *
 * PilasSombrero application
 *
 * @author
 * @version 1.00 2013/10/16
 */

import java.util.Scanner;

public class PilasSombrero
{

   public static void main(String[] args)
   {
          //objeto para lectura datos
          Scanner leer = new Scanner (System.in);

          //declarar pilas P y Q
          Pilas pilaP=new Pilas();
          Pilas pilaQ=new Pilas();

          //variables auxiliares
          int op,valor,tam,coincidente=0;

          System.out.println ("\n\n\t\t\tPILAS SOMBRERO\n\n");

          //ingreso de datos en pilaP

          System.out.println ("\t\tIngreso de datos en la primera Pila\n\n");
          System.out.print ("Desea continuar <1/0>: ");
          op=leer.nextInt();
          while(op==1)
          {
                    System.out.print ("\nIngrese un valor: ");
                    valor=leer.nextInt();
                    pilaP.insertar(valor);
                    System.out.print ("Desea continuar <1/0>: ");
                    op=leer.nextInt();
          }

          //ingreso de datos en PilaQ
```

```
System.out.println ("\n\n\t\tIngreso de datos en la segunda Pila\n\n");
System.out.print ("Desea continuar <1/0>: ");
op=leer.nextInt();
while(op==1)
{
	System.out.print ("\nIngrese un valor: ");
	valor=leer.nextInt();
	pilaQ.insertar(valor);
	System.out.print ("Desea continuar <1/0>: ");
	op=leer.nextInt();
}

//VISUALIZAR LAS PILAS INGRESADAS
System.out.println ("\n\n\tLas pilas ingresadas son .....\n\n");

System.out.println ("\t\t\tPILA P\n\n");
pilaP.recorrer();
System.out.println ("\n\n\t\t\tPILA Q\n\n");
pilaQ.recorrer();


//VERIFICAR PILAS SOMBRERO
//una pila p es sombrero de Q cuando P es nulao cuando tienen los mismos elementos

if(pilaP.vacia()==true && pilaQ.vacia()==false)
{
	System.out.println ("\n\nLa pila P si es sombrero de la Pila Q.....\n\n");
}
else
{
	if(pilaP.vacia()==false && pilaQ.vacia()==true)
	{
		System.out.println ("\n\nLa pila P No es sombrero de la Pila Q.....\n\n");
	}
	else
	{
		if(pilaP.vacia()==true && pilaQ.vacia()==true)
		{
			System.out.println ("\n\nLa pila P No es sombrero de la Pila Q.....\n\n");
		}
		else
		{
			//si las dos pilas tienen el mismo numero de elementos se justifica
			//compararlos sino no es necesario
			if(pilaP.tamanio()==pilaQ.tamanio())
			{
				tam=pilaP.tamanio(); //guardo el tamanio de la pila antes de
extraer
				while(pilaP.vacia()==false)
				{
					if(pilaP.extraer()==pilaQ.extraer())
					{
						coincidente++;
					}
				}
```

```
                                        if(coincidente==tam)
                                        {
                                        System.out.println ("\n\nLa pila P SI es sombrero de la Pila
Q......\n\n");
                                        }
                                        else
                                        {
                                        System.out.println ("\n\nLa pila P No es sombrero de la
PilaQ......\n\n");
                                        }
                                }
                                else
                                {
                                System.out.println ("\n\nLa pila P No es sombrero de la Pila Q......\n\n");
                                }
                        }
                }
        }
   }
}
```

Funcionamiento:

El programa permite determinar si una Pila P es sombrero de otra Pila Q, bajo la siguiente condición: decimos que una pila P es un sombrero de otra pila Q, si todos los elementos de P están en Q, en el mismo orden, y en las posiciones más próximas a la cima. La pila nula se considera un sombrero de cualquier pila.

MANEJO DE CONTENEDORES

Para el siguiente ejercicio se requiere implementar la clase pilas para manejar datos tipo String, tomando como base la clase Pilas anteriormente definida para datos enteros a continuación la clase modificada:

```
public class PilasString
{
        //ATRIBUTOS DE LA CLASE
        //ELEMENTOS DE LA PILA

        String pila[]=new String [10];
        int tope =-1;
        final int MAX = 9;

        //IMPLEMENTAR LAS OPERACIONES CON LAS PILAS

        //VERIFICAR SI LA PILA ESTA VACIA

        public boolean vacia()
        {
                if (tope==-1)
                {
                        return true;
```

```
		}
		else
		{
			return false;
		}
}

//VERIFICAR SI LA PILA ESTA LLENA

public boolean llena()
{
		if (tope==MAX)
		{
			return true;
		}
		else
		{
			return false;
		}
}

//INSERTAR ELEMENTOS

public void insertar(String valor)
{
		if(vacia()==true) //si esta vacia inserto el elemento en a pos 0
		{
			tope=0;
			pila[tope]=valor;
		}
		else
		{
			if(llena()==true) //si esta llena no se inserta nada
			{
				System.out.println ("\nNo se pueden insertar más elementos en la pila\n");
			}
			else //si no esta ni vacia ni llena inserto en tope+1
			{
				tope++;
				pila[tope]=valor;
			}
		}

}

//RECORRER LOS ELEMENTOS DE LA PILA

public void recorrer()
{
		if(vacia()==false)
		{
			for(int i=tope; i>=0; i--)
			{
				System.out.println (pila[i]);
			}
		}
```

```
                else
                {
                        System.out.println ("\nLa pila no tiene elementos\n");
                }
        }

        //EXTRAER ELEMENTOS DE LA PILA

        public String extraer()
        {
                if(vacia()==true)
                {
                        System.out.println ("\nNo se pueden extraer elementos .... la pila está vacia\n");
                        return "";
                }
                else
                {
                        String valor=pila[tope];
                        pila[tope]="";
                        tope --;
                        return valor;
                }
        }

        //METODO PARA CONOCER CUANTOS ELEMETOS TIENE LA PILA

        public int tamanio()
        {
                return(tope+1);
        }

}
```

A continuación la clase principal:

```
/**
 * @(#)PilasContenedores.java
 *
 * Una bodega almacena la mercaderías de las empresas clientes en contenedores, los mismos
 * que están apilados uno a continuación de otro.
 * Si el dueño del contenedor desea retirar su mercadería deberá moverse los contenedores
 * que están arriba de él y almacenarse en otra pila hasta retirarlo
 *
 * @author
 * @version 1.00 2013/10/18
 */

import  java.util.Scanner;
public class PilasContenedores
{

   public static void main(String[] args)
   {
        //declarar el objeto para lectura de datos
```

```
        Scanner leer = new Scanner (System.in);

        //declarar los objetos tipo pila
        PilasString cont=new PilasString();
        PilasString aux=new PilasString();

        //variables extras
        String empresa,auxiliar;
        int op,tam,i;

        //titulo de la aplicación

        System.out.println ("\n\n\t\t\tBODEGA ALMACENERA DE CONTENEDORES\n\n");
        System.out.println ("Guardar contenedores.....\n\n");

        do
        {
                System.out.print ("\nA que empresa le pertenece el contenedor: ");
                empresa=leer.next();
                cont.insertar(empresa);
                System.out.print ("\nDesea ingresar otro contenedor <1/0>: ");
                op=leer.nextInt();
        }while(op==1);

        System.out.println ("\n\n\t\t\tCONTENEDORES EN BODEGA\n\n");
        cont.recorrer();

        System.out.println ("\n\n\t\t\tRETIRO DE CONTENEDORES DE BODEGA\n\n");
        System.out.print ("\n\nA qué empresa pertenece el contenedor que va a retirar: ");
        empresa=leer.next();

        //numero de contenedores existentes en bodega
        tam=cont.tamanio();

        int bandera =0;

        for(i=0; i<=tam-1;i++)
        {
                auxiliar=cont.extraer();
                if(empresa.equals(auxiliar)==true)
                {
                        System.out.println ("\n\nEl contenedor de la empresa "+auxiliar+" ha sido
retirado\n");
                        bandera=1;
                }
                else
                {
                        aux.insertar(auxiliar);
                }
        }

        if(bandera==0)
        {
                System.out.println ("\n\nSu empresa no posee contenedores en esta bodega\n\n");
        }
        while(aux.vacia()==false)
```

```
        {
                cont.insertar(aux.extraer());
        }

        System.out.println ("\n\n\t\t\tMERCADERÍA RESTANTE EN BODEGA\n\n");
        cont.recorrer();

    }
}
```

Funcionamiento:
El programa anterior permite gestionar los contenedores almacenados en una bodega y que se encuentran apilados uno sobre otro, formando una pila.

EJERCICIOS CON COLAS

SIMULACIÓN DE COLAS EN UN BANCO

Para la implementación de este ejercicio, se trabajará con la clase Colas Frente fijo que se implementó anteriormente y que trabaja con datos tipo int, pero será necesario implementar una clase Colas para datos tipo String la misma que se detalla a continuación. Entonces en el ejercicios se agregará dos clases externas y se tendrá la clase principal.

Por favor recuerde entonces agregar la clase ColasFrenteFijo

COLA PARA DATOS TIPO STRING

```
public class ColasString
{

        //DEFINIR LOS ATRIBUTOS DE LA CLASE, ES DECIR LOS ELEMENTOS DE LA COLA

        String cola[]=new String[10];
        int frente=-1;
        int tope=-1;
        final int MAX=9;


        public boolean vacia()
        {
                if(frente==-1 && tope==-1)
                {
                        return true;
                }
                else
                {
                        return false;
                }
        }
```

```
	public boolean llena()
	{
		if(frente==0 && tope==MAX)
		{
			return true;
		}
		else
		{
			return false;
		}
	}

	//INSERTAR ELEMENTOS EN LA COLA

	public void insertar(String valor)
	{
		//verificar si la cola esta vacia
		if(vacia()==true)
		{
			frente=0;
			tope=0;
			cola[tope]=valor;
		}
		else
		{
			if(llena()==true)
			{
				System.out.println ("\n\nNO se pueden insertar elementos.... la cola esta
llena\n\n");
			}
			else
			{
				tope++;
				cola[tope]=valor;
			}
		}

	}

	//RECORRER LOS ELEMENTOS DE LA COLA


	public void recorrer()
	{
		if(vacia()==true)
		{
			System.out.println ("\n\nNo hay elementos para imprimir\n\n");
		}
		else
		{
			for(int i=frente; i<=tope;i++)
			{
				System.out.println (cola[i]);
			}
		}
```

```
	}

	public String extraer()
	{
		String valor="";
		if(vacia()==true)
		{
			System.out.println ("\n\nNo se pueden extraer elementos.....\n\n");
		}
		else
		{
			if(tope==frente)
			{
				valor=cola[frente];
				cola[frente]="";
				frente=-1;
				tope=-1;
			}
			else
			{
				valor=cola[frente];
				cola[frente]="";
				for(int i=frente;i<=tope;i++)
				{
					cola[i]=cola[i+1];
				}
				tope--;
			}
		}
		return valor;
	}
}
```

CLASE PRICIPAL

```
/**
 * @(#)SimulaconColaBancos.java
 *
 * SimulaconColaBancos application
 *
 * @author
 * @version 1.00 2013/10/23
 */
import java.util.Scanner;

public class SimulaconColaBancos
{

   public static void main(String[] args)
   {
	Scanner leer = new Scanner (System.in);

	String nombre;
```

```
int t;
int op;

//declarar los objetos de las clases
ColasString nombres = new ColasString();
ColasFrenteFijo trans = new ColasFrenteFijo();

System.out.println ("\n\n\t\t\tBANCO SOLIDARIO\n\n\n");

do
{
	System.out.println ("\n\nIngrese sus datos ...");
	System.out.print ("\nCliente: ");
	nombre=leer.next();
	nombres.insertar(nombre);
	System.out.println ("\nElija la transaccíon .....");
	System.out.println ("1. Retiros --> 4 minutos");
	System.out.println ("2. Depositos --> 2 minutos");
	System.out.println ("3. Consultas --> 3.5 minutos");
	System.out.println ("4. Actualizacion --> 5 minutos");
	System.out.println ("5. Pagos --> 2 minutos");
	do
	{
		System.out.print ("\nTransaccion: ");
		t=leer.nextInt();
	}while(t<=0 || t>5);
	trans.insertar(t);
	System.out.print ("\n\nDesea ingresar otro cliente <1/0>: ");
	op=leer.nextInt();
}while(op==1);

System.out.println ("\n\n\nClientes en espera .....\n");
nombres.recorrer();


double tespera=0;
double tatencion;

System.out.println ("\n\n\n\t\t\tAtencíon a los clientes\n\n");


do
{
	System.out.println ("\n\nCLIENTE: "+nombres.extraer());
	int x=trans.extraer();
	if(x==1)
	{
		tatencion=4;
	}
	else
	{
		if(x==2)
		{
			tatencion=2;
		}
		else
```

```
                        {
                                if(x==3)
                                {
                                        tatencion=3.5;
                                }
                                else
                                {
                                        if(x==4)
                                        {
                                                tatencion=5;
                                        }
                                        else
                                        {
                                                tatencion=2;
                                        }
                                }
                        }
                }
                System.out.println ("\nSu tiempo de espera fue de: "+tespera+" minutos");
                System.out.println ("\nSu tiempo de atención fue de: "+tatencion+" minutos");
                System.out.println ("\nSu tiempo de permanencia en el banco fue de:
"+(tespera+tatencion)+" minutos");
                tespera=tespera+tatencion;
        }while(nombres.vacia()==false);
   }
}
```

Funcionamiento:
El ejercicio anterior permite simular una cola de un banco mostrándole al usuario su tiempo de espera en cola y su tiempo de atención.

EJERCICIOS CON LISTAS

RESERVAS DE VUELOS

Para el siguiente ejercicio se trabajará con la clase Listas creada anteriormente y que como se pudo ver trabaja para datos tipos String, pero será necesario definir la clase para el tratamiento de Listas con datos enteros.

A continuación la Clase correspondiente:

```
public class ListasInt
{

        //ATRIBUTOS DE LA CLASE
        int lista[]=new int[10];
        int tope=-1;
        final int MAX=9;
```

```
//metodos para implementar operaciones con listas

public boolean lista_vacia()
{
	if(tope==-1)
	{
		return true;
	}
	else
	{
		return false;
	}
}

public boolean lista_llena()
{
	if(tope==MAX)
	{
		return true;
	}
	else
	{
		return false;
	}
}

//metodo para crear en una lista de manera secuencial

public void crear_lista(int valor)
{
	if(lista_vacia())
	{
		tope=0;
		lista[tope]=valor;
	}
	else
	{
		if(lista_llena())
		{
		System.out.println ("\nLa lista esta llena ... no se pueden insertar elementos\n");
		}
		else
		{
			if(tope<MAX)
			{
				tope++;
				lista[tope]=valor;
			}
		}
	}
}

//IMPRIMIR LOS ELEMENTOS DE LA LISTA

public void imprimir()
```

```
{
	if(lista_vacia())
	{
		System.out.println ("\nNo existen elementos en la lista\n");
	}
	else
	{
		System.out.println ("\n\n\t\t\tElementos de la lista \n\n");
		for(int i=0;i<=tope;i++)
		{
			System.out.println ("Elemento "+i+": "+lista[i]);
		}
	}
}

//INSERTAR UN VALOR A PARTIR DE UN POSICION DETREMINADA

public void insertar_posicion(int pos, int valor)
{
	if(lista_vacia()==true && pos==0)
	{
		tope=0;
		lista[tope]=valor;
	}
	else
	{
		if(lista_llena())
		{
			System.out.println ("\n\nNo se puede insertar el elemento .... Lista
llena\n\n");
		}
		else
		{
			if(pos>=0 && pos<=tope)
			{
				for(int i=tope;i>=pos;i--)
				{
					lista[i+1]=lista[i];
				}
				lista[pos]=valor;
				tope++;
			}
			else
			{
				System.out.println ("\n\nLa posicion no es la correcta ...\n\n");

			}
		}
	}
}

//METODOS PARA REALIZAR BUSQUEDAS

//ENVIO EL VALOR A BUSCAR Y RETORNA LA POSICION EN LA QUE LO ENCONTRO

public int buscarPorValor(int valor)
```

```
{
	int i,pos=-1,b=0;
	for(i=0;i<=tope && b==0;i++)
	{
		if(lista[i]==valor)
		{
			pos=i;
			b=1;
		}
	}
	return pos;
}

//BUSCAR A PARTIR DE UNA POSICION Y RETORNAR EL VALOR

public int buscarPorPos(int pos)
{
	int valor;
	int i;
	if(pos>=0 && pos<=tope)
	{
		return lista[pos];
	}
	else
	{
		return -1;
	}
}

//BUSCAR UN ELEMENTO ESPECIFICO EN UNA POSICION DETERMINADA

public boolean buscarElemPos(int valor, int pos)
{
	if(pos>=0 && pos<=tope)
	{
		if(lista[pos]==valor)
		{
			return true;
		}
		else
		{
			return false;
		}
	}
	else
	{
		return false;
	}
}

//BUSCAR PRIMERA OCURRENCIA DE UN ELEMENTO EN LA LISTA

public int primera_ocurrencia(int valor)
{
	return buscarPorValor(valor);
}
```

```
	//CONTAR EL NUMERO DE OCURRENCIAS DE UN ELEMENTO EN LA LISTA

	public int contar_elementos(int valor)
	{
		int i,cont=0;
		if(lista_vacia())
		{
			System.out.println ("\n\nLa lista esta vacia .... no se pueden contar los
elementos\n");
		}
		else
		{
			if(buscarPorValor(valor)!=-1)
			{
				for(i=0;i<=tope;i++)
				{
					if(lista[i]==valor)
					{
						cont++;
					}
				}
			}
		}
		return cont;

	}


	//BUSCAR ULTIMA OCURRENCIA DE UN ELEMENTO EN LA LISTA

	public int ultima_ocurrencia(int valor)
	{
		int i,b=0,pos=-1;

		if(lista_vacia())
		{
			System.out.println ("\n\nNo se puede buscar el elemento.... lista vacia\n");

		}
		else
		{
			if(contar_elementos(valor)>1)
			{
				for(i=tope;(i>=0 && b==0);i--)
				{
					if(lista[i]==valor)
					{
						pos=i;
						b=1;
					}
				}
```

```
                }
                else
                {
                        pos=buscarPorValor(valor);
                }

        }
        return pos;
}

//INSERTAR ELEMENTO A PARTIR DE UN VALOR DETERMINADO

public void insertar_valor(int valorb, int valori)
{

        int pos;

        if(lista_llena())
        {
                System.out.println ("\n\nLa lista esta llena ... no se pueden insertar elementos\n");
        }
        else
        {
                pos=buscarPorValor(valorb);
                if(pos!=-1)
                {
                        for(int i=tope;i>=pos;i--)
                        {
                                lista[i+1]=lista[i];
                        }
                        lista[pos+1]=valori;
                        tope++;
                }
                System.out.println ("\n\nLista resultante.....\n\n");
                imprimir();
        }
}

//ORDENAR LISTA ASCENDENTE Y DESCENDENTEMENTE

public void ordenar_lista(int tipoorden)
{
        int i,j;
        int aux;

        if(tipoorden==1)
        {
                for(i=0;i<=tope-1;i++)
                {
                        for(j=i+1;j<=tope;j++)
                        {
                                if(lista[j]<lista[i])
                                {
                                        aux=lista[j];
                                        lista[j]=lista[i];
                                        lista[i]=aux;
```

```
                                        }
                                }
                        }
                }
                else
                {
                        for(i=0;i<=tope-1;i++)
                        {
                                for(j=i+1;j<=tope;j++)
                                {

                                        if(lista[j]>lista[i])
                                        {
                                                aux=lista[j];
                                                lista[j]=lista[i];
                                                lista[i]=aux;
                                        }
                                }
                        }

                }

        }


        //EDITAR UN ELEMENTO DE LA LISTA

        public void editar(int valor, int nuevo)
        {
                int pos=buscarPorValor(valor);
                if(pos!=-1)
                {
                        lista[pos]=nuevo;
                        System.out.println ("\n\nLista resultante ....\n");
                        imprimir();
                }
                else
                {
                        System.out.println ("\n\nEl valor solicitado no existe ....\n\n");
                }
        }
}
```

CLASE PRINCIPAL

Recuerde agregar la clase para datos tipos String definida anteriormente, en total este programa contendrá 3 clases diferentes:

```
/**
 * @(#)ReservasVuelos.java
 *
 * ReservasVuelos application
 *
 * @author
 * @version 1.00 2013/11/15
```

```
 */

import java.util.Scanner;

public class ReservasVuelos
{

   public static void main(String[] args)
   {

         //OBJETO PARA LECTURA DE DATOS
         Scanner leer = new Scanner (System.in);

         //OBJETOS DE LA CLASE LISTAS
         Listas cedulas=new Listas();
         Listas apellidos = new Listas();
         Listas nombres=new Listas();
         ListasInt reservas = new ListasInt();

         //DECLARAR VARIABLES AUXILIARES
         String ced, nom, ape;
         int tipo,asiento,op;


         //ENCABEZADOS Y TITULOS
         System.out.println ("\n\n\t\t\tAGENCIA DE VIAJES VIAJE SEGURO\n\n");
         System.out.println ("\t\t\tRESERVAS DE VUELO\n\n\n");
         do
         {

                  //**********************DATOS PERSONALES DEL
PASAJEROS****************

                  System.out.println ("DATOS DEL PASAJERO: \n");

                  //VALIDACIÓN DE CEDULA REPETIDAS
                  do
                  {
                           System.out.print ("CEDULA: ");
                           ced=leer.next();
                           if(cedulas.buscarPorValor(ced)!=-1)
                           {
                                    System.out.println ("\nEl pasajero ya esta registrado\n");
                           }
                  }while(cedulas.buscarPorValor(ced)!=-1);

                  System.out.print ("APELLIDOS: ");
                  ape=leer.next();
                  System.out.print ("NOMBRES: ");
                  nom=leer.next();
                  cedulas.crear_lista(ced);
                  apellidos.crear_lista(ape);
                  nombres.crear_lista(nom);

                  //**********************DATOS DE LA RESERVA DE VUELO ***************
```

```
System.out.println ("\n\n\t\t\tELIJA EL TIPO DE PASAJERO\n\n");

System.out.println ("1. Primera Clase \n");
System.out.println ("2. Discapacitado\n");
System.out.println ("3. Fumador\n");
System.out.println ("4. Pasajero comun\n\n");

//VALIDACION DEL TIPO DE PASAJERO
do
{
        System.out.print ("En que categoria desea viajar: ");
        tipo=leer.nextInt();
}while(tipo<1 || tipo>4);

switch(tipo)
{
        case 1:
        {
                System.out.println ("\n\n\t\t\tPASAJEROS DE PRIMERA CLASE\n\n");
                do
                {
                        System.out.print ("Ingrese el numero de asiento <1 .. 6>: ");
                        asiento=leer.nextInt();
                }while(asiento<1|| asiento>6);

                if(reservas.buscarPorValor(asiento)!=-1)
                {
                        System.out.println ("\n\nEse asiento ya esta reservado.....\n");
                }
                else
                {
                        reservas.crear_lista(asiento);
                        System.out.println ("\n\nSu asiento ha sido reservado\n");
                }
                break;
        }

        case 2:
        {
                System.out.println ("\n\n\t\t\tPASAJEROS DISCAPACITADOS\n\n");
                do
                {
                        System.out.print ("Ingrese el número de asiento <7 .. 12>: ");
                        asiento=leer.nextInt();
                }while(asiento<7|| asiento>12);

                if(reservas.buscarPorValor(asiento)!=-1)
                {
                        System.out.println ("\n\nEse asiento ya está reservado.....\n");
                }
                else
                {
                        reservas.crear_lista(asiento);
                        System.out.println ("\n\nSu asiento ha sido reservado\n");
                }
                break;
```

```
                        }
                        case 3:
                        {
                                System.out.println ("\n\n\t\t\tPASAJEROS AREA DE
FUMADORES\n\n");
                                do
                                {
                                        System.out.print ("Ingrese el número de asiento <79 .. 84>: ");
                                        asiento=leer.nextInt();
                                }while(asiento<79|| asiento>84);

                                if(reservas.buscarPorValor(asiento)!=-1)
                                {
                                        System.out.println ("\n\nEse asiento ya está reservado.....\n");
                                }
                                else
                                {
                                        reservas.crear_lista(asiento);
                                        System.out.println ("\n\nSu asiento ha sido reservado\n");
                                }
                                break;
                        }
                        case 4:
                        {
                                System.out.println ("\n\n\t\t\tPASAJEROS COMUNES\n\n");
                                do
                                {
                                        System.out.print ("Ingrese el número de asiento <13 .. 78>: ");
                                        asiento=leer.nextInt();
                                }while(asiento<13|| asiento>78);

                                if(reservas.buscarPorValor(asiento)!=-1)
                                {
                                        System.out.println ("\n\nEse asiento ya esta reservado.....\n");
                                }
                                else
                                {
                                        reservas.crear_lista(asiento);
                                        System.out.println ("\n\nSu asiento ha sido reservado\n");
                                }
                                break;
                        }
                        default:
                        {
                                System.out.println ("\n\nOpcion incorrecta\n\n");
                        }

                }
                System.out.print ("\n\nDesea reservar otro asiento <1/0> : ");
                op=leer.nextInt();

        }while(op==1);

        //NUMERO DE PASAJEROS QUE ESTAN EN LA LISTA
        int x=cedulas.tope;
```

```
        //IMPRESION DE RESULTADOS
        System.out.println ("\n\n\n\t\t\tLISTA DE PASAJEROS Y RESERVAS\n\n");
        System.out.println ("CEDULA \t\t\tAPELLIDOS\t\t\tNOMBRES\t\t\tRESERVA\n\n");

        for(int i=0;i<=x;i++)
        {
                System.out.println
(cedulas.buscarPorPos(i)+"\t\t\t"+apellidos.buscarPorPos(i)+"\t\t\t"+nombres.buscarPorPos(i)+"\t\t\t"+reserva
s.buscarPorPos(i)+"\n");
        }
    }
}
```

Funcionamiento:

El ejercicio permite realizar la reserva de vuelos en función del tipo de asiento que se desea utilizar y de la disponibilidad de los mismos. Considerando que un usuario no puede reservar más de un asiento.

REFERENCIAS

Groussard, T. (2012). *Los fundamentos del lenguaje Java.* Barcelona: Ediotions ENI.

Joyanes Aguilar, L. (2008). *Fundamentos de programación.* Espña: McGraw-Hill.

Orta, C. (s.f.). *Introducción a Java.*

Sánchez, J., Huecas, g., & et.al. (2005). *PROGRAMACIÓN EN JAVA 2.* . España: McGraw-Hill.

Universidad de Cantabria. (2013). *Estructuras de Datos y Algoritmos.* España.

Dentro del presente texto se introduce al lector en el manejo del marco conceptual requerido para dar sus primeros pasos en el desarrollo de programas informáticos en el lenguaje de programación java, así también, se lo orienta en la aplicación de un proceso sistemático que inicia con el análisis del problema, el diseño de la solución, el desarrollo de la aplicación y las pruebas que validan los resultados obtenidos, temas que fortalecen el desarrollo del pensamiento lógico, requisito indispensable para un programador.

Además se presenta un conjunto amplio de ejercicios que van incrementando el nivel de dificultad y abarcan temas como el manejo de sentencias de entrada – salida, sentencias de control, manejo de arreglos, la programación orientada a objetos y el manejo de estructuras de datos, áreas indispensables en el desarrollo de programas en cualquier lenguaje de programación.

ISBN- 978-9942-21-158-3

ISBN 978-9942-21-158-3
9 789942 211583