ITCA FEPADE
TÉCNICOS E INGENIEROS

DIPLOMADO PROGRAMADOR EN JAVA.

Instructor: Giovanni Ariel Tzec Chávez

Java Básico (Modulo 1)

1 Fundamentos del lenguaje Java.

1.1. La tecnología Java.
1.2 Sintaxis.
1.3 Identificadores.

2.Gramática de Java.

2.1. Tipos de datos.
2.2. Conversión de datos.
2.3. Operadores.
2.4. Primera clase de objetos

3. Controles y matrices en Java.

3.1. Control de flujo.
3.2. Bucles.
3.3. Matrices.

Retroalimentando clase 2

- ✓Identificadores y tipos de datos.
- ✓Clase Scanner
- ✓Formatos numéricos y de fecha
- ✓Estructuras selectivas.
- ✓Constantes
- ✓JOptionPane

Contenido

- ✓Estructuras repetitivas (contadores y acumuladores)
- ✓Estructura de datos
- ✓Vectores y matrices.
- ✓Introducción a la modularidad.

Desarrollo de ejercicios

1. Un profesor tiene las notas de sus 10 alumnos y necesita conocer la nota promedio y la nota mayor de todas. Diseñe un programa para ayudarle al profesor.

2. Calcule el aumento de salarios para n empleados de una empresa, bajo el siguiente criterio:

- Si el salario es menor a $1000.oo aumento 12%
- Si el salario está comprendido
- Entre $1000.oo y $2500.o aumento 10%
- Si el sueldo es mayor a $2500.oo aumento 8%
- Cantidad de personas que ganan entre 1000 y 2500, sueldo mayor, sueldo menor.

3.Diseñe un programa que reciba como entrada 24 números reales que representan las temperaturas en todo un día (1 lectura por hora) y calcule: la temperatura promedio, número de datos menores que el promedio y número de datos mayores que 28°.

4.Tomando en cuenta la categoría y el salario de un empleado, diseñe un programa que calcule el aumento correspondiente de acuerdo a la siguiente tabla:

Categoría	Aumento
1	20
2	15
3	10
4	7

Debe imprimirse el nombre, la categoría y el nuevo salario del empleado.

5. En una empresa con varios empleados se necesita obtener cierta información. Por cada empleado se ingresan los siguientes datos: edad, sexo y salario.

Diseñe un programa para calcular lo siguiente:

- Número de hombres
- Número de mujeres
- Número de mujeres que ganen más de $1000.oo
- Número de hombres menores de 40 años que ganan menos de $1000.oo
- Número de empleados mayores de 50 años.

Estructura de datos

En programación, una estructura de datos es una forma particular de organizar datos en una computadora para que pueda ser utilizado de manera eficiente. Diferentes tipos de estructuras de datos son adecuadas para diferentes tipos de aplicaciones, y algunos son altamente especializados para tareas específicas.

Las estructuras de datos son un medio para manejar grandes cantidades de datos de manera eficiente para usos tales como grandes bases de datos y servicios de indización de internet.

Hay tres tipos de estructuras de datos clásicas en java:

Arreglos: Unidimensionales y bidimensionales**.**

Mapas: utiliza e implementa la interfaz Map.

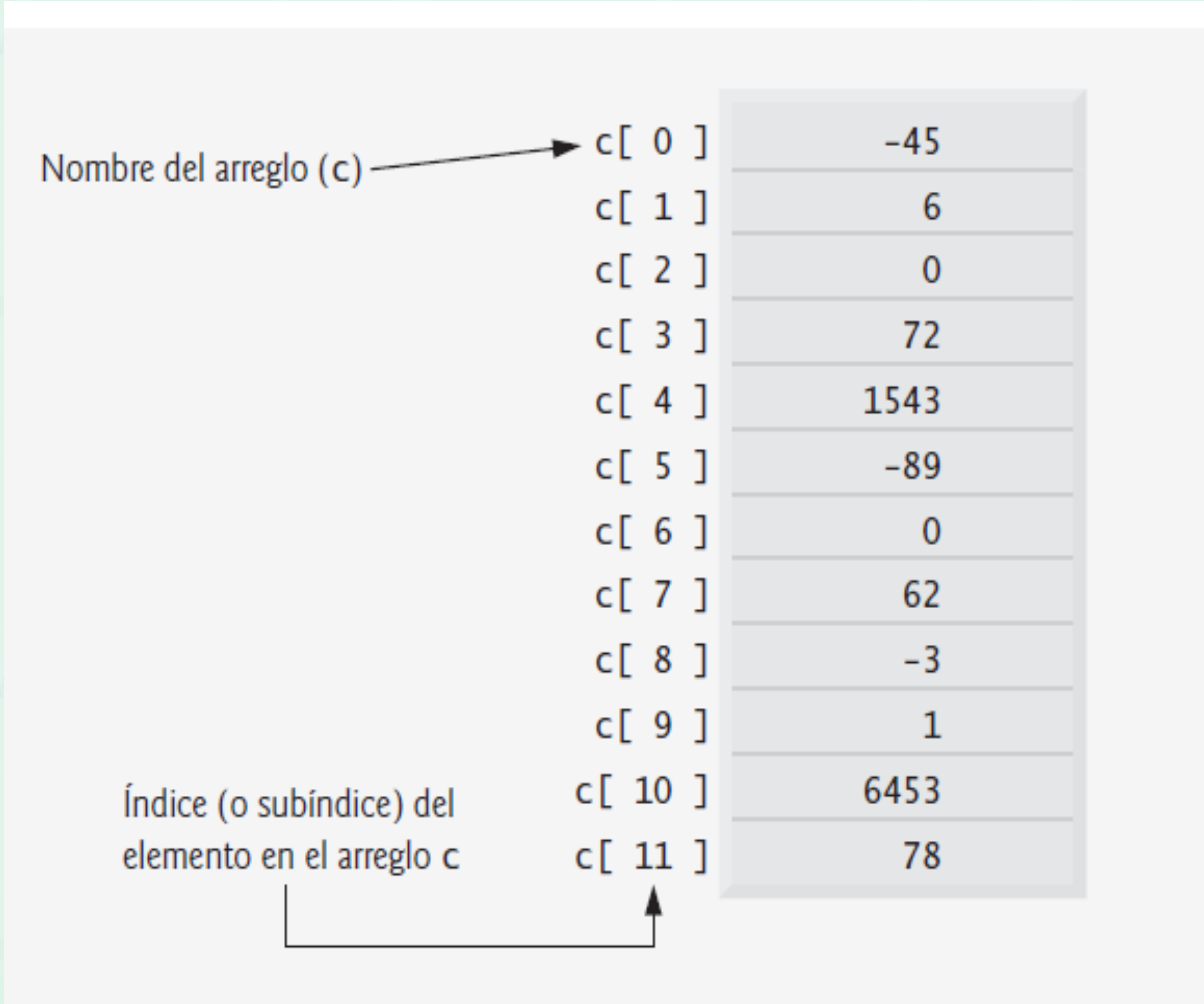
Colecciones: implementan la interfaz collection; entre ellas la interfaz Set y la interfaz List.

Arreglos

Son estructuras de datos que consisten de elementos de datos relacionados, del mismo tipo. Los arreglos son entidades de longitud fija; conservan la misma longitud una vez creados, aunque puede reasignarse una variable tipo arreglo de tal forma que haga referencia a un nuevo arreglo de distinta longitud.

- Un arreglo se puede considerar simplemente como una tabla, con una sola fila de información (también podemos visualizar una tabla como una sola columna de información). Esta podría ser una tabla de números, de cadenas de texto o de cualquier otra cosa.

Ejemplo


Nombre del arreglo (c)
c[0] -45
c[1] 6
c[2] 0
c[3] 72
c[4] 1543
c[5] -89
c[6] 0
c[7] 62
c[8] -3
c[9] 1
c[10] 6453
c[11] 78
Índice (o subíndice) del elemento en el arreglo c

Sintaxis de los arreglos unidimensionales

int c[ ] //Declara la variable de un arreglo.

c= new Int[12];//Crea el arreglo

int c[ ]= new int[12];

Mostrar ejemplo.

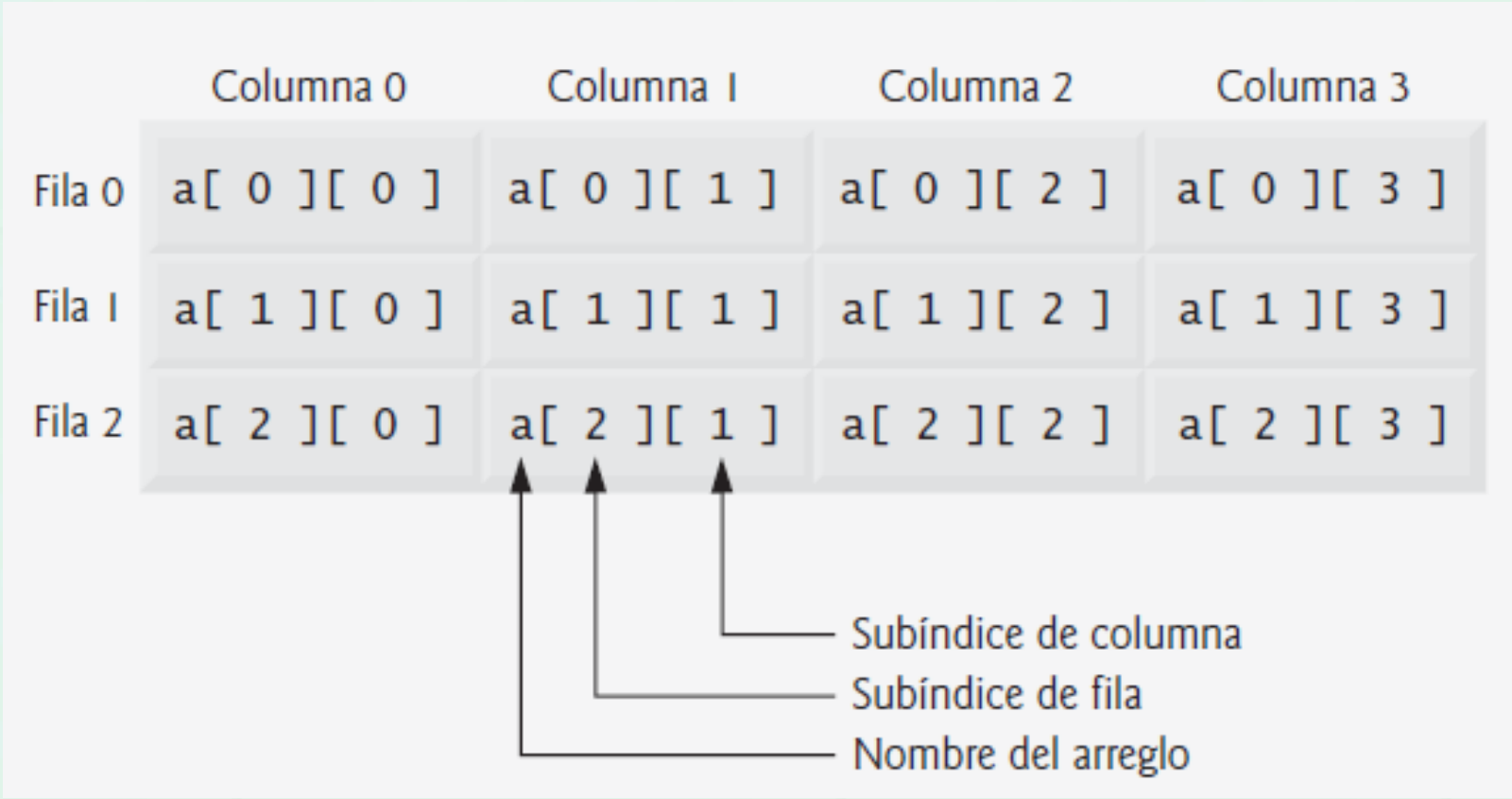
Ejemplo de vector utilizando método

```java
package vectorfunciones;
import java.util.*;
/**
 *
 * @author Giovanni Tzec
 */
public class VectorFunciones {

    public static void main(String[] args) {
        int vec[]= new int[5];
        int i;
        Scanner leer=new Scanner(System.in);
        for(i=0;i<vec.length;i++)
        {
            System.out.println("Ingrese un numero: " + i);
            vec[i]=leer.nextInt();
        }
        parImpar(vec);

    }
```

```java
    public static void parImpar(int vec[])
    {
        int i, contapares,contaimpares;
        contapares=0;
        contaimpares=0;
        for(i=0;i<vec.length;i++)
        {
            if(vec[i]%2==0)
            {
                contapares=contapares+1;
            }
            else
            {
               contaimpares=contaimpares+1;
            }
        }
        System.out.println("Cantidad de numeros pares: "+contapares);
        System.out.println("Cantidad de numeros impares: "+contaimpares);
    }

}
```

Arreglos multidimensionales

- Los arreglos multidimensionales, se utilizan con frecuencia para representar tablas de valores, las cuales consisten en información ordenada en fi las y columnas. Para identificar un elemento especifico de una tabla, debemos especificar dos subíndices. Por convención, el primero identifica la fila del elemento y el segundo su columna. Los arreglos que requieren dos subíndices para identificar un elemento especifico se llaman arreglos bidimensionales (los arreglos multidimensionales pueden tener mas de dos dimensiones).


Columna 0
Columna 1
Columna 2
Columna 3
Fila 0
a[0][0]
a[0][1]
a[0][2]
a[0][3]
Fila 1
a[1][0]
a[1][1]
a[1][2]
a[1][3]
Fila 2
a[2][0]
a[2][1]
a[2][2]
a[2][3]
Subíndice de columna
Subíndice de fila
Nombre del arreglo

Sintaxis

Java no soporta los arreglos multidimensionales directamente, pero se definen mediante arreglos de arreglos unidimensionales.

int b[ ][ ] = new int[3][4];

***tipoArreglo nombreArreglo*[][] = new *tipoArreglo*[*numFilas*][*numColumnas*];**

Desarrollo de ejercicios con vectores y matrices.

1. Desarrollar una aplicación que lea 3 notas de 3 estudiante de un grupo (Elaborarlo por matriz) e imprima el promedio de cada uno, almacenando los promedios en un vector.

Calcular el promedio de la nota2

Calcular la nota mayor y menor de la nota 3

Crear la aplicación con al menos 3 métodos.

2. Se tiene un vector de 50 elementos se necesita saber si todos son positivos o negativos.

Para ello se le pide que diseñe un programa que imprima “CIERTO” si todos son positivos, “FALSO” si todos son negativos y, “MIXTO” si el vector tiene elementos positivos y negativos.

3. En un vector de 25 elementos, se desea buscar un dato que será leído. Diseñe un programa que imprima el mensaje: “VALOR ENCONTRADO”, si en efecto el valor buscado ya se encuentra entre los elementos, y la posición o subíndice donde se encontró la primera vez.

Si el valor no se encuentra, se debe sustituir el valor menor por el buscado, imprimiendo el elemento que sale del vector y la posición que ocupaba.

Modularidad

La modularidad es la técnica conocida como “Divide y vencerás”, se encarga de dividir un problema en secciones pequeñas, para posteriormente desarrollar cada sección y para luego unir las soluciones en una sola solución a toda la problemática.

Elementos importantes al aplicar modularidad.

Variables locales.

Variables globales.

Funciones.

Métodos.

Mostrar ejemplo de métodos con parámetros y sin parámetros.

Mostrar funciones con parámetros y sin parámetros.

Definir static en métodos y funciones.

Manipulando matrices con métodos

```java
package matrizfunciones;
import java.util.*;
/**
 *
 * @author Giovanni Tzec
 */
public class MatrizFunciones {

    public static void main(String[] args) {
        int matriz[][]=new int[2][2];
        int i,j;
        Scanner leer=new Scanner(System.in);
        for(i=0;i<matriz.length;i++)
        {
            for(j=0;j<matriz.length;j++)
            {
                System.out.println("Ingrese un valor en la siguiente posición: "+i+j);
                matriz[i][j]=leer.nextInt();
            }
        }
        imprimirdiagonal(matriz);

    }
```

```java
    public static void imprimirdiagonal(int matriz[][])
    {
     int i,j;
        for(i=0;i<matriz.length;i++)
       {
           for(j=0;j<matriz.length;j++)
           {
               if(i==j)
               {
                   System.out.println("Valores de la diagonal: " + matriz[i][j]);
               }
           }
       }
    }
}
```

Bibliografía.

Piensa en Java 2 Edicion Spanish

Aprenda Java Desde Cero.

Preguntas y respuestas