Java

Clases

Referencia a objetos this

- Referencia especial que utilizada dentro de un método de cualquier clase se refiere a la instancia actual
  - Permite parámetros con igual nombre que los atributos
  - Posibilita la llamada a otros constructores

Llamado a otro constructor

- La llamada a un constructor también puede realizarse desde dentro de otro constructor de su misma clase y debe ser siempre la primera instrucción.

Ejemplo

```
public class fecha{
  public Fecha(int d, int m, int a) {
    this();  //llamada al constructor sin parámetros

                  …….

  }
}
```

Ejemplo

```
class Persona {
    String nombre;
    String apellido;
    String direccion;

    public Persona() {
        this.nombre = "No información";
        this.apellido = "No información";
        this.direccion = "No información";
    }

    public Persona(String nombre) {
        this(); // debe ser la primera línea
        this.nombre = nombre;
    }

    public Persona(String nombre, String apellido,
String direccion) {

        this.nombre = nombre;
        this.apellido = apellido;
        this.direccion = direccion;
    }}
```

```
public String toString() {
        return "Persona = " + /*this.*/nombre + " " + apellido + " - Dir: " + direccion;
    }

    public static void main(String[] args) {
        Persona p1 = new Persona();
        Persona p2 = new Persona("Edu");
        Persona p3 = new Persona("Pepe", "Garcia", "Gran Via 14");
        System.out.println(p1.toString());
        System.out.println(p2.toString());
        System.out.println(p3.toString());
    }
}
```

Constructor copia

- Se puede crear un objeto a partir de otro de su misma clase escribiendo un constructor llamado **constructor copia**
- Este constructor copia los atributos de un objeto existente al objeto que se está creando.
- Tiene un solo argumento, el cual es una referencia a un objeto de la misma clase que será desde el que queremos copiar.

Ejemplo

```
Fecha f1 = new Fecha(1,1,2011);    //se invoca al constructor con parámetros
Fecha f2 = new Fecha(f1);   //se invoca al constructor copia
        //constructor copia de la clase Fecha
    public Fecha(Fecha f) {
        dia = f.dia;
        mes = f.mes;
        año = f.año;
    }
```

Ejemplo

```
class PersonaC {
        String nombre;
        String apellido;
        String direccion;

        public PersonaC() {
                nombre="No identificado";
                apellido="No identificado";
                direccion=     "No identificado";
                }
        // constructor copia
        public PersonaC(PersonaC p) {
                /*this.*/nombre = p.nombre;
                this.apellido = p.apellido;
                this.direccion = p.direccion;
        }
```

```
    public String toString() {
        return "Persona = " + nombre + " " + apellido + " - Dir: " + direccion;
    }

    public static void main(String[] args) {
        PersonaC p1 = new PersonaC();
        PersonaC p4 = new PersonaC(p1);
        System.out.println(p1.toString());
        System.out.println(p4.toString());
    }
}
```

Variables static de la clase y Constructor

- Pertenece a todas las instancias de la clase.
- Puede estar como pública o como privada.
- Las variables de tipo static son, en algunos aspectos, parecidas a las variables globales de algunos lenguajes.
- Las instancias de la clase comparten la variable static

Ejemplo

```
class MiClase{
  //declaramos e inicializamos la variable estática
  static int contador = 0;
  public MiClase(){
    //Se modifica el valor en el constructor
    contador += 1;
  }

  public static void main (String[] args){
    MiClase k = new MiClase();
    new MiClase();
    new MiClase();
    System.out.println("El número de clases son: "+MiClase.contador);
  }
}
```

Ejemplo que no puede ser ejecutado

```
class MiClase{
   //declaramos e inicializamos la variable de instancia
  int contador = 0;
  public MiClase(){
    //Se modifica el valor en el constructor
    contador += 1;
  }
  public static void main (String[] args){
    new MiClase();
    new MiClase();
    new MiClase();
    System.out.println("El número de clases son: "+contador);
  }
}
MiClase.java: 11: non-static variable contador cannot be referenced from static context
  System.out.println("El número de clases son: "+contador);
                                                 ^
  1 error
```

Nota

- La JVM no sabe de qué objeto de *MiClase* estás tratando de imprimir su variable *'contador'*.
- El problema es que el método *main()* en sí es estático y no está corriendo sobre un objeto en particular de la clase sino directamente en la clase.
- **Un método estático no puede acceder a ningún componente (método o variable) no estático.**

Ejemplo-Ya lo hicieron

```
public class Docente{
        private String nombre, apellido, tipo;
        private  int horas;

       public Docente(String nombre, String apellido, String tipo, int horas){
                this.nombre = nombre;
                this.apellido = apellido;
                this.tipo = tipo;
                this.horas = horas;

       }
        public String getNombre(){ return nombre;}
        public String get Apellido (){ return apellido;}
        public String get Tipo (){ return tipo;}
        public int get Horas (){ return horas;}
```

```
    public void setNombre(String nombre){this.nombre = nombre;}
    public void set Apellido (String apellido){this.apellido = apellido;}
    public void set Tipo (String tipo){this.tipo = tipo;}
    public void set Horas (int horas){this.horas = horas;}


    public String getNombreCompleto(){
      return nombre + " " + apellido;
}
```

```
    public double getSueldoBruto (){
        if(tipo.equals(“ciencia”))
            return 3 * horas;
        else
            return 5 * horas;
    }
public double getDescuento (){
        return 0.10* getSueldoBruto ();
}


public double getNeto (){
        return getSueldoBruto () - getDescuento ();
}
```

```
public static void main(String[] args) {
        Docente d1 = new Docente(“JUAN”,”MENDEZ”,”ARTES”,25);
        System.out.println(p4. getNombreCompleto());
        System.out.println(d1.getNeto);
    }
}
```

Docente

Arreglos

- Los arreglos pueden ser multidimensionales.
- El tamaño no puede cambiar durante la ejecución.
- Se crean utilizando la palabra reservada new.
- Primero, se declara una variable array de tipo deseado.

  Tipo nombrevar[ ]; o Tipo[ ] nombre var;

  Ejemplo:

  int dia_mes [ ]; o int [ ] dia_mes;

- Segundo, se asigna memoria con el operador new.

  Var_array=new tipo[longitud];

  Ejemplo

  dia_mes = new int [12]; //arreglo de 12 enteros

Acceso a los Arreglos

- Para acceder a los elementos de una arreglo

  nom_areglo[posición];

  Ejemplo

  dia_mes[0]=28; //asigna el 28 a la posición 0

  Ejemplo

```
String array[] = new String[10];  //se declara un arreglo y se asigna su tamaño en una misma línea
```

Atributo length

- Devuelve el número de elementos del arreglo

nom_arreglo.length

Ejemplo

```
char array[];
array = new char[10];


for(int x=0;x<array.length;x++)
    System.out.printnln(array[x]);
```

Matrices

Para declarar y crear

```
int matriz[][];
matriz = new int[3][2];
```

Ejemplo

```
int matriz[][];
matriz = new int[2][3];
 matriz[2][3]={{3,5,6}, {9,2,6}}; //sin las 2 instrucciones anteriores
for (int fil=0; fil < matrix.length; fil++) { //matrix.length=no. Filas de la matriz
  for (int col=0; y < matriz[fil].length; col++) {//matrix[fil].length=no. Columnas de la fila fil
      System.out.println (matriz[fil][col]);
  }
}
```

Matrices irregulares

- En Java se pueden crear arrays irregulares en los que el número de elementos de cada fila es variable.
- Es obligatorio indicar el número de filas.
- Por ejemplo:

```
int [][] m = new int[3][];
```

A cada fila se le puede asignar un número distinto de columnas:

```
m[0] = new int[3];
m[1] = new int[5];
m[2] = new int[2];
```

Ejemplo

```
import java.util.*;
 class Clasearreglos {


 public static void main( String args[] ) {
   double a[] = { 12, 23.5, 15, 7, 8.9 };
   float x [] = new float[5];
   double total = 0;
   int i;
   for (i=0; i<x.length; i++)
      /*total*/ x[0] += a[i];
      System.out.println( "La media es:" );
      System.out.println( /*total*/ x[0] / a.length );
 }
}
```

Ejemplo

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

    private Scanner teclado;
    private int[] sueldos;
    public ClaseSueldos()
    {
        teclado=new Scanner(System.in);
        sueldos=new int[5];
        for(int f=0;f<sueldos.length;f++) {
            System.out.print("Ingrese valor ");
            sueldos[f]=teclado.nextInt();
        }
    }
public void imprimir() {
        for(int f=0;f<sueldos.length;f++)
            System.out.println(sueldos[f]);
             }
public static void main(String[] ar) {
        ClaseSueldos op=new ClaseSueldos();
         op.imprimir();
        }
 }// fin de la clase
```

Ejemplo

```
import java.util.Scanner;
public class ArrayDeNombres {
        public static void main(String arg[ ]) {
             int i;
             boolean resp;
             Scanner teclado;
             teclado=new Scanner(System.in);
            String[ ] nombre = new String[4];
            nombre[0] = "Luis";
            nombre[1] = "María";
            nombre[2] = "Carlos";
            nombre[3] = "Jose";
            //nombre[4] = "Ismael";   Error:No existe esta variable array de índice 4
      for(i=0; i<nombre.length;i++){
        System.out.println("Te llamas " + nombre[i]+"?");
        resp=teclado.nextBoolean();
       System.out.println("dices "+resp);
    }
}
}
```

Clase String

Método	*Cometido*
length()	Devuelve la longitud (número de caracteres) de la cadena
charAt (int pos)	Devuelve el carácter que hay en una cierta posición
toLowerCase()	Devuelve la cadena convertida a minúsculas
toUpperCase()	Devuelve la cadena convertida a mayúsculas
substring(int desde, int cuantos)	Devuelve una subcadena: varias letras a partir de una posición dada
replace(char antiguo, char nuevo)	Devuelve una cadena conun carácter reemplazado por otro
trim()	Devuelve una cadena sin espacios de blanco iniciales ni finales
startsWith(String subcadena)	Indica si la cadena empieza con una cierta subcadena
endsWith(String subcadena)	Indica si la cadena termina con una cierta subcadena

Método	*Cometido*
indexOf(String subcadena, [int desde])	Indica la posición en que se encuentra una cierta subcadena (buscando desde el principio, a partir de una posición opcional)
lastIndexOf(String subcadena, [int desde])	Indica la posición en que se encuentra una cierta subcadena (buscando desde el final, a partir de una posición opcional)
valueOf(objeto)	Devuelve un String que es la representación como texto del objeto que se le indique (número, boolean, etc.)
concat(String cadena)	Devuelve la cadena con otra añadida a su final (concatenada) También se pueden concatenar cadenas con "+"
equals(String cadena)	Mira si las dos cadenas son iguales (lo mismo que "= =")
equals-IgnoreCase(String cadena)	Comprueba si dos cadenas son iguales, pero despreciando las diferencias entre mayúsculas y minúsculas
compareTo(String cadena2)	Compara una cadena con la otra (devuelve 0 si son iguales, negativo si la cadena es "menor" que cadena2 y positivo si es "mayor").

Práctica 7

1. Hacer un programa que almacene 40 caracteres, posteriormente debe contar e imprimir cuántos de estos son vocales y cuántos de estos son dígitos.

2. Crea una clase con el método main donde declares una variable **de tipo array** de Strings que contenga los doce meses del año, en minúsculas y declarados en una sola línea. A continuación declara una variable mesSecreto de tipo String, y hazla igual a un elemento del array (por ejemplo mesSecreto = mes[9]. El programa debe pedir al usuario que adivine el mes secreto y si acierta mostrar un mensaje y si no pedir que vuelva a intentar adivinar el mes secreto.

Un ejemplo de ejecución del programa podría ser este:

- Adivine el mes secreto. Introduzca el nombre del mes en minúsculas: febrero
- No ha acertado. Intente introduciendo otro mes: agosto
- No ha acertado. Intente introduciendo otro mes: octubre
- ¡Ha acertado!

Tarea 8

1. Investigar cómo funciona los métodos length, compareTo de java y traer 1 programa utilizando cada uno de los métodos.
2. Hacer un programa que pida al usuario por teclado una frase y que pase sus caracteres a un array de caracteres. Puedes hacerlo con o sin métodos de String.
3. Investigar cómo se declaran y definen los arreglos bidimensionales en java.
4. Hacer dos programas que utilicen arreglos bidimensionales