

JAVA



UTILIZACIÓN DE SUBPROGRAMAS Y CLASES DE USO COMÚN

1. Introducción



1.- Introducción

- Hasta ahora hemos visto un programa como una lista de instrucciones que le indican a la máquina que hacer.
- Con la programación orientada a objetos un programa se convierte en un conjunto de objetos que “dialogan” entre sí para realizar las distintas tareas programadas.
- Por ejemplo si realizamos un programa para una entidad bancaria, esta **entidad** será un objeto que se relacionará con otros objetos tales como los objetos **clientes**, los objetos **cuentas bancarias**, los objetos **sucursales**, los objetos **empleados**...

1.- Introducción

- ¿Qué es entonces una clase?
- Pongamos un ejemplo: pensad en un molde con el que hacemos flanes. El molde será la clase y los flanes los objetos. Luego, cada flan, tendrá sus propios atributos como la cantidad de leche, cantidad de azúcar, número de huevos, cantidad de vainilla...
- Para definir un objeto de la clase Flan sería así:

```
Flan flanvainilla; // Declaro el objeto flanvainilla de la clase Flan
```

```
flanvainilla = new Flan(); //Asigno al objeto flanvainilla las características de la clase
```

- Lo más común es verlo de la siguiente forma:

```
Flan flanvainilla = new Flan();
```

1.- Introducción

- Veamos otro ejemplo: Pensad en un esqueleto para la fabricación de portátiles.
- Este esqueleto sería la clase y los ordenadores portátiles que salgan a partir de él serían los objetos.
- Además, estos objetos tendrán una serie de características (**atributos**) tales como el **color**, **marca**, **procesador**, **tamaño del disco duro**, **cantidad de memoria RAM**, **capacidad de batería**, **peso**...
- Estos objetos también podrán realizar una serie de acciones (**métodos**) tales como **encenderse**, **apagarse**, **cargar el Sistema Operativo**...

1.- Introducción

```
package ordenadorportatil;
/**
 * @author OLG
 */
public class OrdenadorPortatil {

    private String marca;
    private int peso;
    private boolean encendido=false; //Mal. Nunca se debe inicializar un atributo. Se debe utilizar un constructor. Lo veremos más adelante.

    public void encenderOrdenador(){
        if (encendido==true){
            System.out.println("El ordenador ya estaba encendido");
        }
        else{
            encendido=true;
        }
    }

    public void apagarOrdenador(){
        if (encendido==false){
            System.out.println("El ordenador ya estaba apagado");
        }
        else{
            encendido=false;
        }
    }

    public void establecerMarca (String mar){
        marca=mar;
    }

    public void establecerPeso (int pes){
        peso=pes;
    }

    public void obtenerEstado(){
        System.out.println("El estado del ordenador es el siguiente:");
        System.out.println("Marca: "+marca);
        System.out.println("Peso: "+pes);
        if (encendido ==true){
            System.out.println("El ordenador está encendido");
        }
        else{
            System.out.println("El ordenador está apagado");
        }
    }
}
```

7

PROGRAMACIÓN

1.- Introducción

OrdenadorPortatil - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default confi... | [Icons]

Files | Test.java | OrdenadorPortatil.java

Source Packages | ordenadorportatil | OrdenadorPortatil | Test.java | Libraries

```
1 package ordenadorportatil;
2 /**
3  * @author OLG
4  */
5 public class Test {
6     public static void main(String[] args) {
7         OrdenadorPortatil miOrdenador = new OrdenadorPortatil();
8         OrdenadorPortatil ordenadorInsti = new OrdenadorPortatil();
9
10        miOrdenador.establecerMarca("Asus");
11        ordenadorInsti.establecerMarca("TTL");
12
13        miOrdenador.establecerPeso(1600);
14        ordenadorInsti.establecerPeso(1900);
15
16        miOrdenador.encenderOrdenador();
17
18        miOrdenador.obtenerEstado();
19        ordenadorInsti.obtenerEstado();
20    }
21 }
```

8

PROGRAMACIÓN

EJERCICIOS

- **Ejercicio 01.- (OPTATIVO)** Diseña una clase Coche que contenga los siguientes atributos privados:
 - marca (de tipo cadena)
 - modelo (de tipo cadena)
 - color (de tipo cadena)
 - velocidad (de tipo entero)
 - motorEncendido (de tipo booleano inicializado a false)

9

PROGRAMACIÓN

EJERCICIOS

- Además contendrá los siguientes métodos públicos:
 - establecerMarca.
 - establecerModelo.
 - establecerColor.
 - arrancarCoche (pone motorEncendido a true y velocidad a 10)
 - apagarCoche (pone motorEncendido a false y velocidad a 0)
 - acelerarCoche (aumenta en 20 la velocidad)
 - frenarCoche (disminuye 6 la velocidad)
 - obtenerEstado (muestra el valor de los atributos)

10

PROGRAMACIÓN

EJERCICIOS

- Para probar el funcionamiento de la clase Coche, crea otra clase llamada Test (dentro del mismo paquete) que contenga el método main donde se creen dos objetos de la clase Coche: *miCoche* y *cochePadre*. Luego realiza las siguientes operaciones:
 - Establece el modelo, marca y color, primero en el objeto *miCoche* y luego en *cochePadre*.
 - Arranca *miCoche* y luego arranca *cochePadre*.
 - Acelera 5 veces *miCoche*.
 - Frena 2 veces *miCoche*.
 - Acelera 3 veces *cochePadre*.
 - Apaga *cochePadre*.
 - Muestra el estado de *miCoche* y luego el estado de *cochePadre*.

11

PROGRAMACIÓN

JAVA

ESTRUCTURAS DE CONTROL DE FLUJO

2. Clases



2.- Clases

- Las clases son las *plantillas* para crear objetos.
- Clase es un tipo definido por el usuario que describe los atributos y los métodos de los objetos que se crearán a partir de la misma.
- La definición de una clase consta de dos partes: el nombre de la clase precedido por la palabra reservada **class**, y el cuerpo de la clase encerrado entre llaves:

```
class NombreClase {
    ... cuerpo de la clase ...
}
```

13

PROGRAMACIÓN

2.- Clases

- Veamos en más detalle el cuerpo de una clase:

```
[acceso] class NombreClase {
    [acceso] [static] [tipo] atributo1;
    [acceso] [static] [tipo] atributo2;
    [acceso] [static] [tipo] atributo3;
    .....
    [acceso] [static] [tipo] método1(listaDeArgumentos) {
        ... código del método ...
    }
    .....
}
```

14

PROGRAMACIÓN

2.- Clases

- **[acceso]** va a determinar el alcance de la visibilidad del elemento al que se refieren (clase, atributo o método)
- **[acceso]** podrá ser de tipo **public** (visible para cualquier clase), **protected** (visible para la clase propia y las heredadas), **private** (solo visible para la clase propia). Si no indicamos nada por defecto el acceso será **friendly** (visible para todas las clases del paquete)

15

PROGRAMACIÓN

2.- Clases

zona	<i>private</i> (privado)	sin modificador (friendly)	<i>protected</i> (protegido)	<i>public</i> (público)
Misma clase	X	X	X	X
Subclase en el mismo paquete		X	X	X
Clase (no subclase) en el mismo paquete		X		X
Subclase en otro paquete			X	X
No subclase en otro paquete				X

16

PROGRAMACIÓN

2.- Clases

- **[tipo]** se refiere al tipo de datos, tanto para atributos como para métodos (*char*, *int*, *double*...). Los métodos también pueden ser del tipo *void* (vacío) lo que significa que no devuelven ningún valor (el método no contiene la sentencia **return**)
- **[static]** lo utilizaremos cuando queramos que un atributo o un método sea genérico para todos los objetos de la clase. Veamos un ejemplo:

17

PROGRAMACIÓN

2.- Clases

```
package cliente;
/**
 * @author DLG
 */
public class Cliente {

    private static int numeroClientes = 0;
    private String nombre;
    private int numeroCuenta;

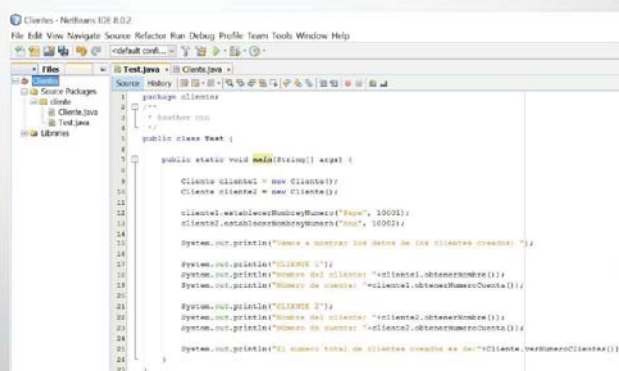
    public static void aumentarClientes() {
        numeroClientes++;
    }

    public static int verNumeroClientes() {
        return numeroClientes;
    }

    public void establecerNombreYNumero(String nom, int num) {
        nombre = nom;
        numeroCuenta = num;
        aumentarClientes();
    }

    public String obtenerNombre() {
        return nombre;
    }

    public int obtenerNumeroCuenta() {
        return numeroCuenta;
    }
}
```



```
Clientes - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
package cliente;
/**
 * @author DLG
 */
public class Cliente {

    public static void main(String[] args) {
        Cliente cliente1 = new Cliente();
        Cliente cliente2 = new Cliente();

        cliente1.establecerNombreYNumero("Juan", 10001);
        cliente2.establecerNombreYNumero("Ana", 10002);

        System.out.println("Vamos a mostrar los datos de los clientes creados:");

        System.out.println("CLIENTE 1");
        System.out.println("Nombre del cliente: " + cliente1.obtenerNombre());
        System.out.println("Número de cuenta: " + cliente1.obtenerNumeroCuenta());

        System.out.println("CLIENTE 2");
        System.out.println("Nombre del cliente: " + cliente2.obtenerNombre());
        System.out.println("Número de cuenta: " + cliente2.obtenerNumeroCuenta());

        System.out.println("El número total de clientes creados es de: " + Cliente.verNumeroClientes());
    }
}
```

18

PROGRAMACIÓN

EJERCICIOS

- **Ejercicio 02.- (OPTATIVO)** Diseña una clase Curso que contenga los siguientes atributos privados:
 - nombre (de tipo cadena)
 - numeroHoras (de tipo entero)
- Además dispondrá de un atributo estático llamado numeroDeCursos de tipo entero que lo utilizaremos para contar los objetos que vamos creando.
- La clase Curso contendrá los siguientes métodos públicos:
 - establecerNombreyHoras.
 - obtenerNombre.
 - obtenerHoras.

19

PROGRAMACIÓN

EJERCICIOS

- Además, la clase Curso, dispondrá de dos métodos estáticos (para trabajar con el atributo estático):
 - sumarCursos
 - verNumeroCursos
- Para probar el funcionamiento de la clase Curso, crea otra clase llamada Test (dentro del mismo paquete) que contenga el método main, donde se creen dos objetos de la clase Curso: *curso1* y *curso2*. Luego realiza las siguientes operaciones:
 - Establece el nombre y el número de horas, primero en el objeto *curso1* y luego en *curso2*.
 - Muestra los datos de los 2 cursos creados.
 - Por último, muestra el número de cursos creados almacenados en el atributo estático de clase *numeroDeCursos*.

20

PROGRAMACIÓN