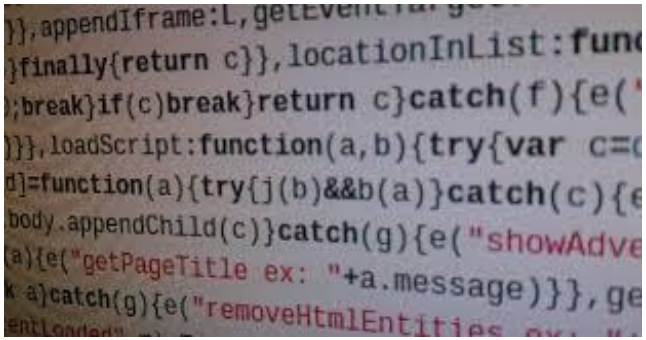


UT 7

INTRODUCCIÓN POO.

Utilización de Objetos

SEGUNDA PARTE

IES JUAN DE LA CIERVA
DPTO. INFORMÁTICA

CONTENIDOS DE LA PRESENTACIÓN

Objetivos

Esta unidad propone la utilización de objetos ya definidos en el propio lenguaje, la creación por el usuario de objetos sencillos, así como las propiedades de estos.

Contenidos

1. Utilización de métodos.
2. Utilización de propiedades.
3. Parámetros y valores devueltos.
4. **Ámbito de las variables.**
5. **Librerias de Objetos, clases envoltorio.**
6. **Sobrecarga de métodos**
7. **Constructores.**
8. Paso por valor y por referencia
9. Destrucción de objetos y liberación de memoria.

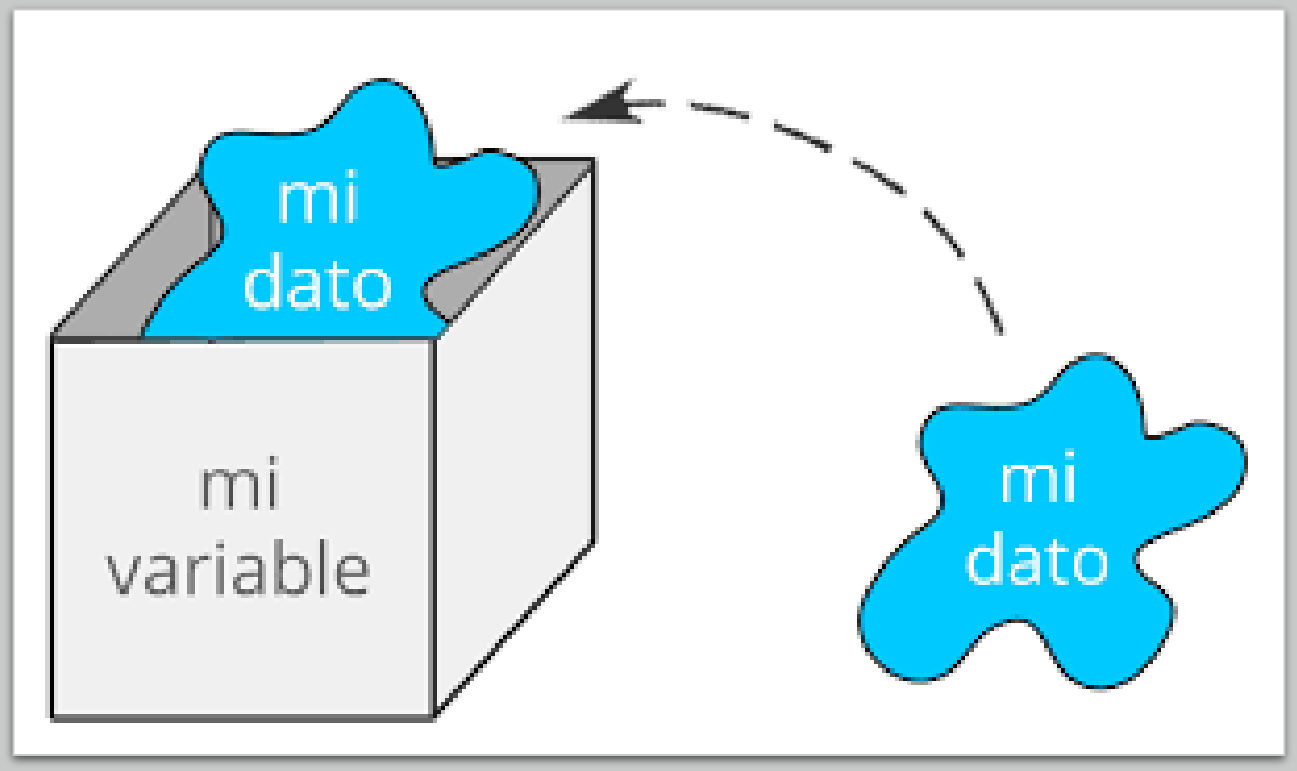
MÉTODOS

Método sin valor de salida

```
public void metodo(…) {
      //…
}
```

Método sin valores de entrada

```
public int metodo() {
      return this.valor;
}
```



ÁMBITO DE LAS VARIABLES

Ámbito de las Variables

- **Variables locales (automáticas)**
  - declaradas dentro del cuerpo de un método
  - visibles sólo dentro del cuerpo de un método
  - se mantienen en una pila

- **Variable de instancia**
  - declaradas dentro de una clase pero fuera de los métodos
  - se crean por cada objeto instanciado

- **Variables de clase (estáticas)**
  - declaradas dentro de una clase pero fuera de los métodos
  - declaradas con el modificador static
  - compartidas por todas los objetos instanciados

Ámbito de las variables

Acceso a campos y métodos de otra clase.

```
public class AccountTest {
      public static void main(String[] args){

        1) SavingsAccount sa0001 = new SavingsAccount();
      2) {sa0001.name = "Damien";
          sa0001.deposit(1000);
}
```

```
public class SavingsAccount {
      public String name;
      public double balance;

      public void deposit(int x){
          balance += x;
      }
}
```

Ámbito de las variables

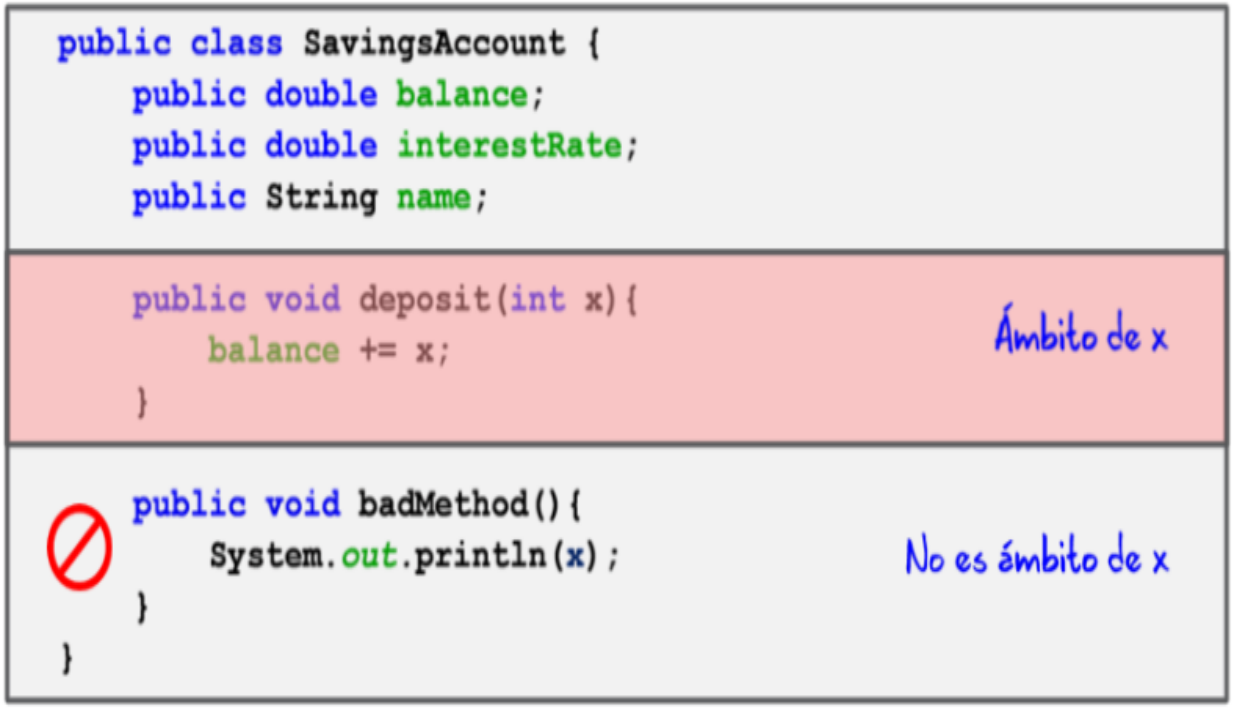
Se puede acceder a las variables de instancia desde cualquier parte de la clase. Esto incluye los métodos.

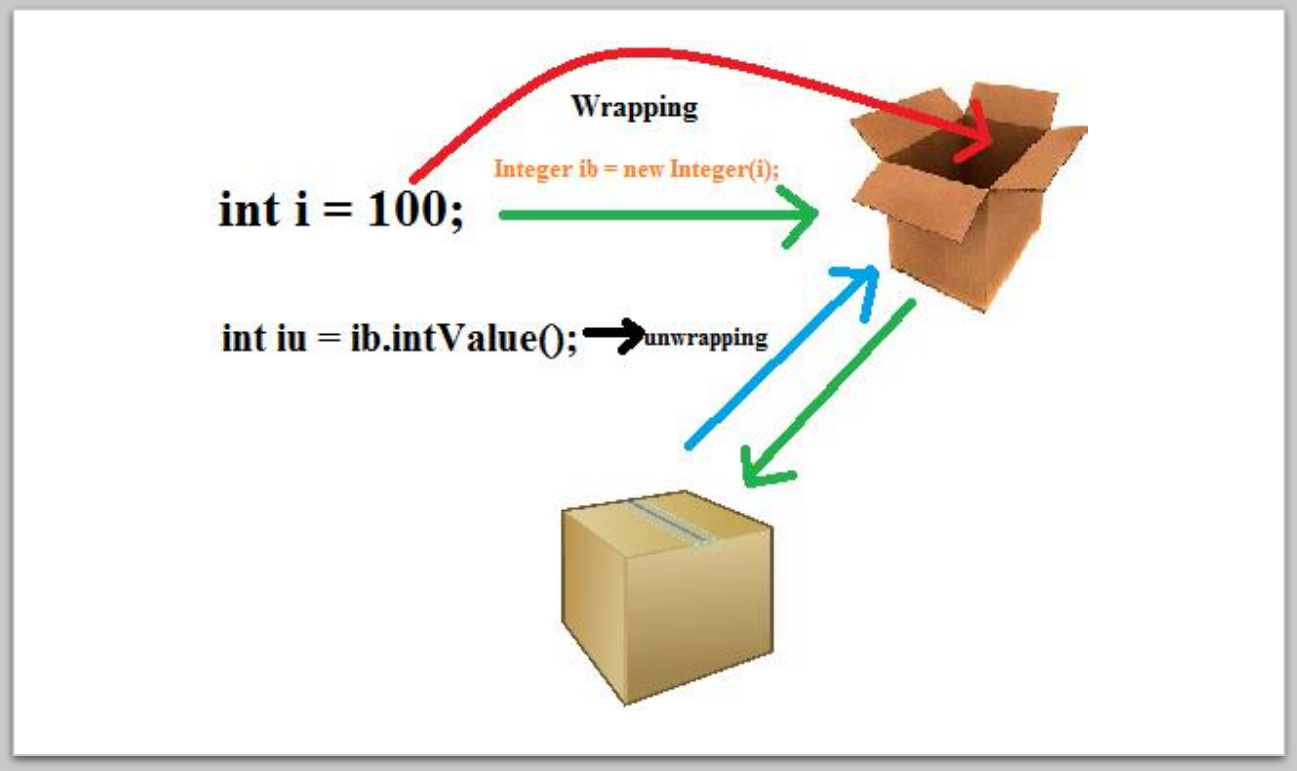
```
public class SavingsAccount {
    public double balance;
    public double interestRate;
    public String name;

    public void displayCustomer(){
        System.out.println("Customer: "+name);
        System.out.println("Balance: " +balance);
        System.out.println("Rate: "    +rate);
    }
}
```

Ámbito de las variables

NO se puede acceder a las variables declaradas dentro de un método desde fuera de él.





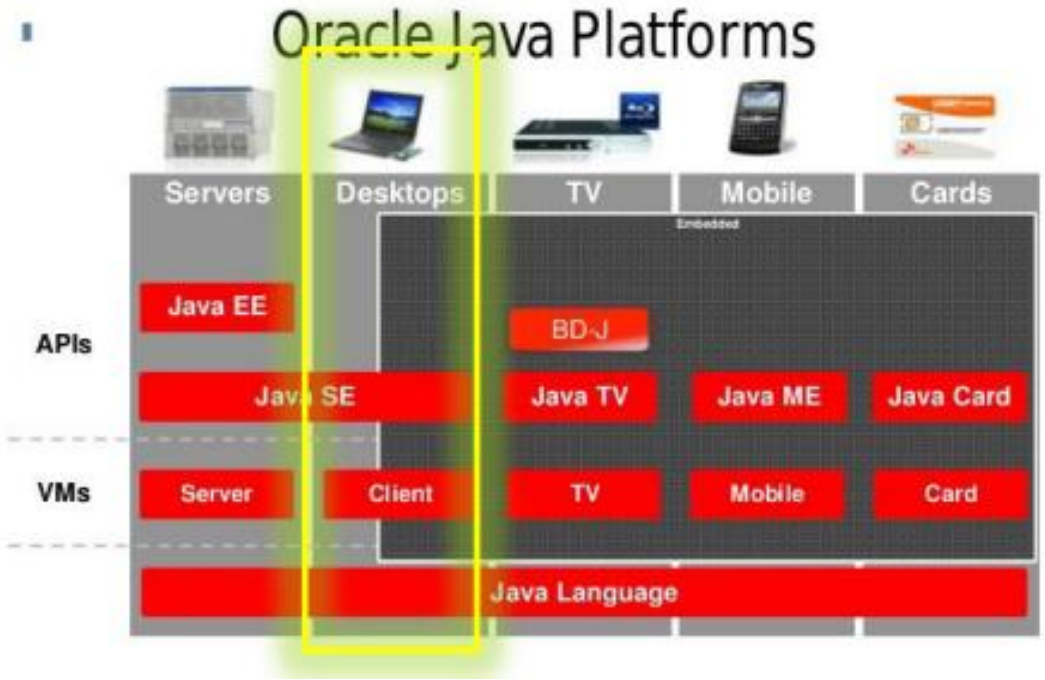
LIBRERIAS DE OBJETOS Y CLASES ENVOLTORIO

Librerias en Java

¿CUANTAS CLASES INCLUYE JAVA SE?

> 4000

Java pone a nuestra disposición más de 4000 clases en su versión 8



Librerias en Java

DOCUMENTACIÓN DE JAVA

- Cada versión de Java tiene publicada una ayuda online para consultar la documentación de cada una de sus clases, en formato HTML.
- Se le conoce como API JAVA DOCS (o también como API).

https://docs.oracle.com/javase/8/docs/api/index.html

Clases envoltorio

- Java tiene una clase para cada uno de los tipos de datos primitivos.

Tipo de dato primitivo	Clase envoltorio	Tipo de dato primitivo	Clase envoltorio
boolean	Boolean	int	Integer
char	Character	long	Long
byte	Byte	float	Float
short	Short	double	Double

Ejercicio: Clases envoltorio

Partiendo del la clase ClaseEnvoltorio.java que se proporciona y utilizando la Api de Java, documenta el programa explicando brevemente la operación que realiza cada método de las clases envoltorio utilizadas.

SOBRECARGA DE MÉTODOS

Métodos: Sobrecarga

- La **firma de un método** es la combinación del nombre y los tipos de los parámetros o argumentos.

  La **sobrecarga de métodos** es la creación de varios métodos con el mismo nombre pero con diferente lista de tipos de parámetros.

  Java utiliza el número y tipo de parámetros para seleccionar cuál definición de método ejecutar.

  Java diferencia los métodos sobrecargados con base en el número y tipo de parámetros o argumentos que tiene el método y **no por el tipo que devuelve**.

Métodos: Sobrecarga

```
1   Ejemplo
2
3   /* Métodos sobrecargados */
4   int calculaSuma(int x, int y, int z){
5       ...
6   }
7   int calculaSuma(double x, double y, double z){
8       ...
9   }
10
11  /* Error: estos métodos no están sobrecargados */
12  int calculaSuma(int x, int y, int z){
13      ...
14  }
15  double calculaSuma(int x, int y, int z){
16      ...
17  }
```

CONSTRUCTORES

CONSTRUCTORES

- Hasta ahora hemos invocado al **constructor por defecto de Java** para instanciar objetos.

Vehiculo coche1 = new Vehiculo();

```
class DemoConstructor {
     public static void main (String args[]) {
            MiClase C1 = new MiClase();
            MiClase C2 = new MiClase();
            System.out.println(C1+”    “+C2);
    }
}
```

CONSTRUCTORES

Comprueba qué ocurre si accedo a los atributos de un objeto antes de asignarles valores.

Si los campos **no se han inicializado**, adquieren un **valor por defecto**.

Tipo de Dato	Valor por Defecto
boolean	False
int	0
double	0,0
Cadena	null
Cualquier tipo de objeto	null

CONSTRUCTORES

Es conveniente **evitar utilizar elementos no inicializados**.
Posible excepción **NullPointerException**

¿Cómo podríamos evitar utilizar tantas líneas de código para inicializar las variables de instancia de los objetos?

Ejercicio

Utilizando la clase vehículo, construye un método para asignar valor a los atributos de un objeto.

Pruébalo instanciándolo dos objetos de la clase vehículo y asignándoles valores.

```
Vehiculo coche1=new Vehiculo();
Vehiculo coche2=new Vehiculo();
coche1.asignarValores(“Renault”,”Space”,7,50,17,5);
coche2.asignarValores(“Lexus”,”F Sport”,2,66,18.8);
```

Constructores

CONSTRUCTORES

- Método especial, invocado con el operador **new**.
- Se ejecuta exclusivamente en el momento de creación de un objeto.
- Sirva para **inicializar valores**.
- Normalmente **public**.
- Con o sin argumentos.
- Puede haber varios en una misma clase.
- Eclipse nos ayuda a generarlos.

CONSTRUCTORES: Estructura

- **Son como cualquier otro método, con alguna excepción:**
  - **No tiene tipo de devolución (ni siquiera void)**
  - **Tiene el mismo nombre que la clase.**
  - **Es public.**
  - **Cuando escribimos un constructor propio el de por defecto de Java deja de estar disponible.**

Modifica el método asignarValores de la clase Vehículo para convertirlo en un constructor y pruébalo.

CONSTRUCTORES: Estructura

La instanciación del objeto ahora quedaría de la siguiente forma.

```
Vehiculo coche1=new Vehiculo((“Renault”,”Space”,7,50,17,5);

Vehiculo coche2=new Vehiculo (“Lexus”,”FSport”,2,66,18.8);
```

Palabra clave this

¿Qué ocurriría en este método?

```
public class Preso {
    public String nombre;

     public mostrarNombre(String nombre){
            System.out.println(nombre);
    }

}
```

Palabra clave this

this es una referencia al objeto actual.

```
public class Preso {
    public String nombre;

     public mostrarNombre(String nombre){
            System.out.println(nombre);
            System.out.println(this.nombre);
    }

}
```

Palabra clave this

```java
// Constructor con algunos parámetros
public Persona(String nombre, String apellidos) {
    this.nombre = nombre;
    this.apellidos = apellidos;
}

public Persona(String nombre, String apellidos, int edad)
    this(nombre, apellidos);
    this.edad = edad;
}
```

Ejemplo I

```
public class EjemploRepaso1 {
    int a = 22;

    public static void main (String []Args) {
        Ambitos ejemplo = new Ambitos();
        ejemplo.metodo();
    }

    public void metodo(){
        int a = 33:
        System.out.println("Variable local a ="+a);
        System.out.println("Campo a = "+this.a);
    }
}
```

Ejemplo II

```
public class EjemploRepaso2 {
    int a = 22;

    public static void main (String []Args) {
        Ambitos ejemplo = new Ambitos();
        ejemplo.metodo(11);
    }

    public void metodo(int a){
        System.out.println("Parametro a ="+a);
        System.out.println("Campo a = "+this.a);
    }
}
```

GENERAR CONSTRUCTORES CON ECLIPSE