

# Introducción a Java

//Parte 2

IT BOARDING

**BOOTCAMP**



# Índice



**01** Clases y objetos

**03** Visibilidad

**02** Métodos

**04** Encapsulamiento

IT BOARDING

**BOOTCAMP**

INTRODUCCIÓN A JAVA

# Programación Orientada a Objetos

IT BOARDING

**BOOTCAMP**

¿La programación orientada a objetos  
puede/debe reemplazar a la programación  
estructurada?

IT BOARDING

**BOOTCAMP**

# // Clases y objetos

Clases: (...) la abstracción del mundo real y (...) bla bla bla ...

Nada de eso: una **clase** es un **tipo de dato definido por el programador**.

Un **objeto** es una **variable**, cuyo tipo de dato es una **clase**.

IT BOARDING

BOOTCAMP





# Encapsulamiento

Las clases encapsulan los algoritmos, y proveen una ***interface amigable*** para el usuario (que será otro programador).

Por ejemplo:

- El control remoto de una televisión
- La llave que pone en marcha el auto
- Los botones que inician y detienen el horno a microondas



## Veamos un ejemplo

La clase Numero podría encapsular algunas de las funciones que desarrollamos durante la práctica anterior.



```
public class Numero
{
    private int valor;

    public boolean esPar(){return
valor%2==0;}

    public boolean esPrimo()
    {
        // ...
    }
}
```

```
Numero n = ???;

boolean par = n.esPar();
boolean primo =
n.esPrimo();
```

# Constructor y métodos de acceso e instancias



Las clases tienen un método especial llamado «constructor» que utilizaremos para crear (o instanciar) los objetos de la clase.

```
public class Numero
{
    private int valor;

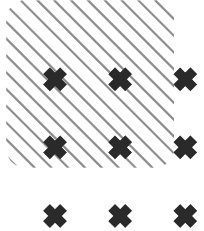
    public void setValor(int v){ valor = v; }
    public int getValor(){ return valor; }

    // ...
}
```

```
Numero n = new Numero();

n.setValor(5);
boolean par = n.esPar();
boolean primo = n.esPrimo();
```





# NullPointerException

Los objetos son **punteros**. Si no los instanciamos, apuntan a **null**.

```
Numero n;  
n.setValor(5);  
boolean nEsPar = n.esPar();  
boolean nEsPrimo = n.esPrimo();
```

```
Número m;  
m.setValor(6);  
boolean mEsPar = m.esPar();  
boolean mEsPrimo = m.esPrimo();
```

# Constructores explícitos y variables de instancia

Podemos **programar explícitamente un constructor** para recibir los valores iniciales que deben tomar las variables de instancia.

```
public class Numero
{
    private int valor;

    public Numero(int v)
    {
        this.valor = v;
    }

    // ...
}
```

```
Numero n = new Numero(5);
boolean nEsPar = n.esPar();
boolean nEsPrimo = n.esPrimo();
```

```
Número m = new Numero(6);
boolean mEsPar = m.esPar();
boolean mEsPrimo = m.esPrimo();
```



# Constantes y variables de clase (static)

Las variables de clase son únicas (y compartidas) por todas las instancias,

```
public class Numero
{
    public static final double PI=3.141592654;
    public static final double E=2.71828;

    private int valor;

    // ...
}
```

```
System.out.println(Numero.PI);
System.out.println(Numero.E);
```

```
Numero n = new Numero(10);
```

```
// ...
```



## Métodos de clase (static)

Son aquellas que pueden trabajar sin acceder a ninguna variable de instancia. Su valor de retorno depende únicamente de sus parámetros.

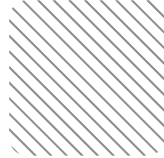
```
public class Numero
{
    private int valor;

    public static int sumar(int a,int b)
    {
        return a+b;
    }

    // ...
}
```

```
int c =
Numero.sumar(2,3);
```





# Diseño de los métodos de una clase

Son aquellas que pueden trabajar sin acceder a ninguna variable de instancia. Su valor de retorno depende únicamente de sus parámetros.

```
public class Numero
{
    // :
    public static int sumar(int a,int b){...}
    public int sumar(int a){...}
    public Numero sumar(Numero n){...}
    // :
}
```

```
int c = Numero.sumar(2,3);
Numero n = new Numero(c);
Numero m = new Numero(8);
Numero x = n.sumar(m);
```



# Clases utilitarias

```
package java.lang;

public class Math
{
    public static int abs(int a){...}
    public static double log(int a){...}
    public static double pow(int a,int b){...}
    public static double sqrt(int a){...}
    // :
}
```



**¿Dudas? ¿Preguntas?**



# Gracias.

IT BOARDING

**BOOTCAMP**

