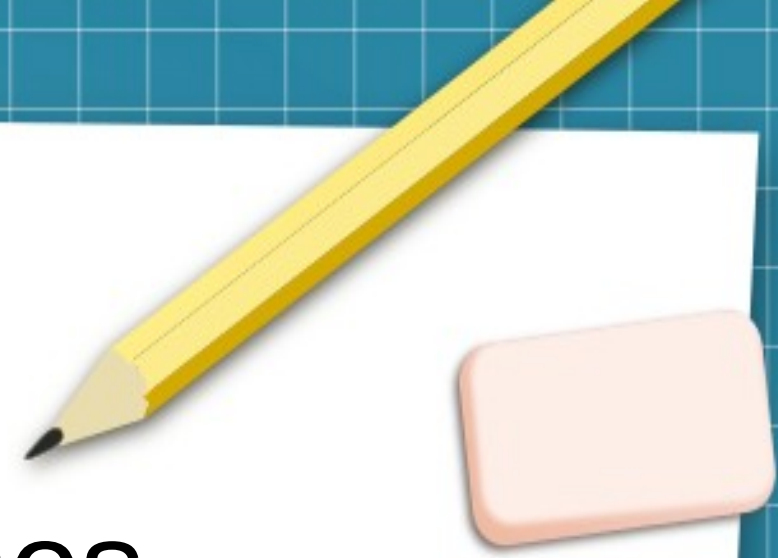


# PОО: Programación orientada a objetos

Bootcamp Java -Roshka



# Objetos en Java



Si consideramos el mundo real, podemos encontrar muchos objetos a nuestro alrededor. Todos estos objetos tienen un estado y un comportamiento.

Por ejemplo:





# Objetos en Java



Si comparamos un objeto de Java con uno del mundo real, estos tienen muchas similitudes.

Los objetos tienen:

- **Estados**
- **Comportamientos**

Los estados de un objeto en software son los campos

Y los comportamientos se muestran a través de métodos

# Clases en Java



Las clases son un esquema o plano de los objetos.

Así tenemos que los objetos son instancias de las clases.

# Clases en Java



Una clase puede contener:

- Variables / propiedades
- Métodos
- Constructores
- Clases e interfaces



```
public class Puppy {  
    // propiedad  
    int puppyAge;  
  
    // constructor  
    public Puppy(String name) {  
        System.out.println("Name chosen is :" + name );  
    }  
  
    // metodos  
    public void setAge( int age ) {  
        puppyAge = age;  
    }  
  
    public int getAge( ) {  
        System.out.println("Puppy's age is :" + puppyAge );  
        return puppyAge;  
    }  
  
    public static void main(String []args) {  
        Puppy myPuppy = new Puppy( "tommy" );  
  
        myPuppy.setAge( 2 );  
  
        myPuppy.getAge( );  
  
        System.out.println("Variable Value :" + myPuppy.puppyAge );  
    }  
}
```

# Modificadores de accesos en Java



Los modificadores de accesos en Java sirven para definir los **niveles de accesos** de las variables o métodos

# Modificadores de accesos en Java



En Java tenemos 4 modificadores:

- default
- public
- private
- protected



## Modificador: default

El modificador de acceso predeterminado también se denomina paquete privado, lo que significa que los miembros están visibles dentro del mismo paquete pero no se puede acceder a ellos desde otros paquetes.





```
package roshka.bootcamp;  
  
public class SuperPublic {  
    static void defaultMethod() {  
        ...  
    }  
}
```



```
package roshka.bootcamp;

public class Public {
    public Public() {
        SuperPublic.defaultMethod(); // disponible en el mismo package
    }
}
```

defaultMethod es accesible en otras clases del mismo paquete



## Modificador: public

Si agregamos la palabra clave pública a una clase, método o propiedad, la pondremos a disposición de todo el mundo, es decir, todas las demás clases en todos los paquetes podrán usarla. Este es el modificador de acceso menos restrictivo





```
package roshka.bootcamp;
```


```
public class SuperPublic {
```

```
    public static void publicMethod() {
```

```
        ...
```

```
    }
```

```
}
```



```
package roshka.bootcamp.otro;

import roshka.bootcamp.SuperPublic;

public class AnotherPublic {
    public AnotherPublic() {
        SuperPublic.publicMethod(); // Disponible en todas partes
    }
}
```

publicMethod es accesible en todos los paquetes



## Modificador: private

Se puede acceder a cualquier método, propiedad o constructor con la palabra clave privada solo desde la misma clase. Este es el modificador de acceso más restrictivo y es fundamental para el concepto de encapsulación. Todos los datos se ocultarán del mundo exterior.



```
package roshka.bootcamp;

public class SuperPublic {
    static private void privateMethod() {
        ...
    }

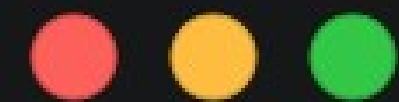
    private void anotherPrivateMethod() {
        privateMethod(); // disponible solo en la misma clase
    }
}
```

Modificador: protected

Si declaramos un método, propiedad o constructor con la palabra clave protected, podemos acceder al miembro desde el mismo paquete (como con el nivel de acceso privado del paquete) y además desde todas las subclases de su clase, incluso si se encuentran en otros paquetes.







```
package roshka.bootcamp;
```

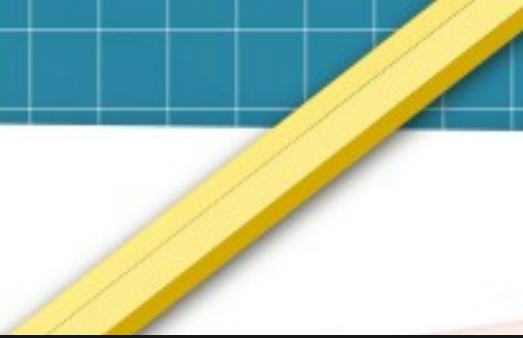
```
public class SuperPublic {
```

```
    static protected void protectedMethod() {
```

```
        ...
```

```
    }
```

```
}
```



```
package roshka.bootcamp.otro;

import roshka.bootcamp.SuperPublic;

public class AnotherSubClass extends SuperPublic {
    public AnotherSubClass() {
        SuperPublic.protectedMethod(); // disponible en subclase
    }
}
```


protectedMethod es accesible en todas las subclases (sin importar el paquete)

# Modificadores de accesos en Java

Es una buena práctica utilizar el nivel de acceso más restrictivo posible para cualquier miembro para evitar el uso indebido. Siempre debemos usar el modificador de acceso privado a menos que haya una buena razón para no hacerlo. El nivel de acceso público solo debe usarse si un miembro es parte de una API.



# Resumen modificadores



Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
default	Y	Y	N	N
private	Y	N	N	N

# Constructores



El constructor de una clase es un método estándar para inicializar los objetos de esa clase.

Se invoca automáticamente cuando **new** crea un objeto de esa clase.

Los constructores se declaran en el momento de definir la clase.



```
class A {  
    int x, y;  
    A() { x=0; y=0; } // el constructor  
    ...  
}
```



# Constructores



Se pueden colocar varios constructores. Durante la creación de un objeto, se invoca aquel que calza con los argumentos dados:



```
class A {  
    int x, y;  
    A() { x=0; y= 0; }  
    A(int valorX, int valorY)  
    { x=valorX; y=valorY; }  
    A(A objetoA)  
    { x= objetoA.x; y= objetoA.y; }  
    ...  
}
```

```
A a1= new A();  
a1.Print();    // 0 0  
A a2= new A(1,2);  
a2.Print();    // 1 2  
A a3= new A(a2);  
a3.Print();    // 1 2
```