Carrera de Java Programmer SE8

Módulo 1 : Fundamentos de Java

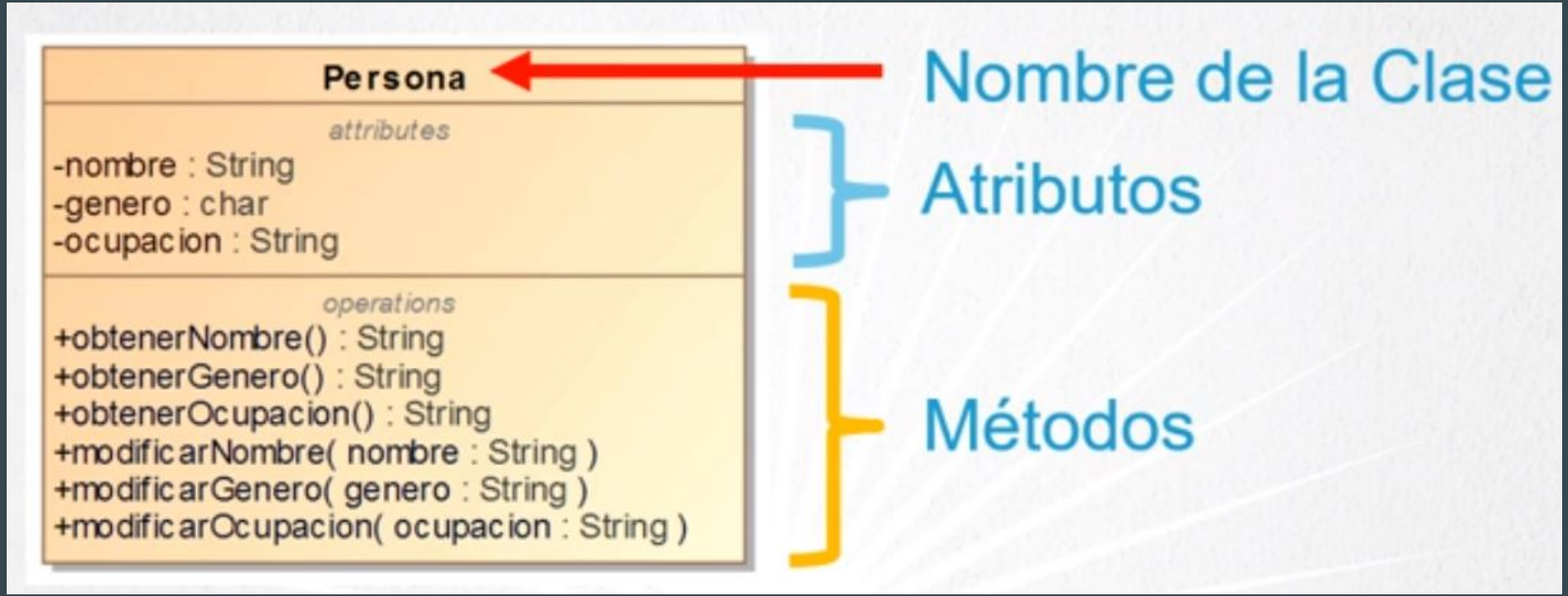
Contenido

- Abstracción
- Encapsulamiento
- Paquetes:
  - Import
  - Import estáticos
  - Paquetes más importantes en Java
- Herencia
- Polimorfismo
- Interfaces
- Excepciones:
  - Manejo de excepciones
  - Excepciones más comunes
  - Stacktrace en Java
  - Creación de nuestras excepciones
- Documentación con JavaDoc
  - Etiquetas

Abstracción

Resaltar la parte más representativa de algo, ignorando detalles para centrarse en lo principal.

Diagrama General de una Clase



Encapsulamiento

Es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa, es decir, evitando el acceso a datos por cualquier otro medio distinto a los especificados.

Modificadores de Acceso

Modificador	Descripción
public	Los elementos pueden ser accedidos desde cualquier clase o instancia sin importar el paquete o procedencia de ésta.
private	Los elementos pueden ser accedidos únicamente por la misma clase.
protected	Permite acceso a los elementos desde la misma clase, clases del mismo paquete y clases que hereden de ella (incluso en diferentes paquetes).
default	Permite que tanto la propia clase como las clases del mismo paquete accedan a dichos componentes.

Ejemplo

```java
public class PruebaEncapsulamiento{//Clase1

    public static void main(String[] args) {
        Persona p1 = new Persona("Juan");
        //Marca error,
        //no se puede acceder directamente un atributo privado desde otra clase
        p1.nombre = "Pedro";
        //Si es posible acceder a un método o atributo publico desde otra clase
        p1.obtenerNombre();
    }
}
```

```java
class Persona {//Clase2

    private String nombre; //Uso de private en un atributo

    public Persona(String nombre) { //Uso de public en un método
        this.nombre = nombre;
    }

    public String obtenerNombre() { //Uso de public en un método
        return nombre;
    }
}
```

Setters / Getters

```java
public class PruebaEncapsulamiento {

    public static void main(String[] args) {
        //Creamos el objeto
        Persona p1 = new Persona();
        //Modificamos el atributo nombre
        p1.setNombre("Juan");
        //Accedemos al atributo nombre
        System.out.println("Nombre:" + p1.getNombre());
    }
}
```

```java
class Persona {
    //Atributo privado
    private String nombre;
    //Método publico para acceder al atributo nombre
    public String getNombre(){
        return nombre;
    }
    //Método publico para modificar al atributo nombre
    public void setNombre(String nombre){
        this.nombre = nombre;
    }
}
```

Ejemplo

Paquetes

- Organización de clases.
- Evitan conflictos entre nombres de clases.
- Limitan acceso a sus clases.
- Utiliza la palabra reservada “package” seguido del nombre del paquete.
  - Usar minúsculas.
  - Usar nombres cortos para uno mejor identificación.
- Estructura:
  - paquete.clase / paquete.subpaquete.clase
  - paquete.clase.metodo / paquete.subpaquete.clase.metodo

Import

- Se utiliza la palabra reservada “import” para utilizar la funcionalidad de un paquete externo.
- “import” debe ser utilizada sobre la declaración de la clase.
- No utiliza espacio en memoria.
- Usos:
  - Específico
  - Generico (*)
- Import estático:
  - import static paquete.clase.metodo

Import Estático

```java
package com.gm; //Definición del paquete

public class Utileria {
    public static void imprimir(String s){
        System.out.println("Imprimiento mensaje: " + s);
    }
}
```

```java
//Importamos el método estático a utilizar
import static com.gm.Utileria.imprimir;

public class EjemploPaquetes {

    public static void main(String[] args) {
        imprimir("Hola");
    }
}
```

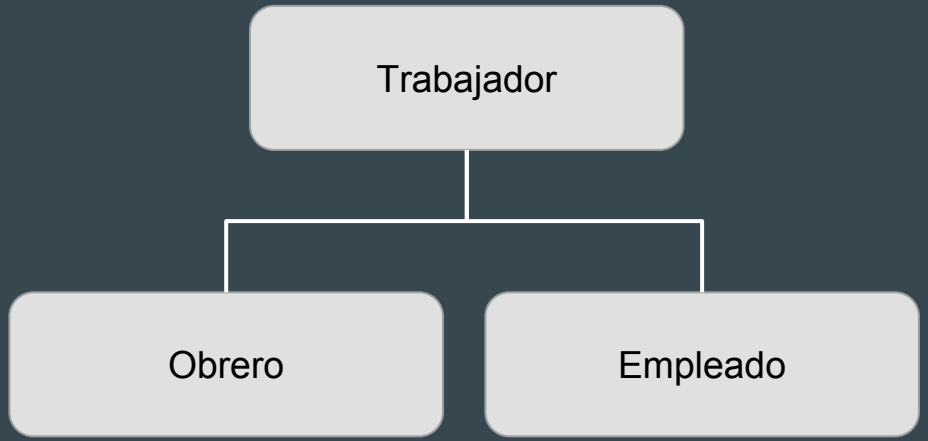
Paquetes más importantes en Java

Paquete	Contenido
java.lang	Contiene clases esenciales, se importa implícitamente sin necesidad de un sentencia import
java.util	Contiene las clases de utilería más comunes (Scanner).
java.io	Clases que definen distintos flujos de datos (input/output).
java.net	Se usa en combinación con las clases del paquete java.io para leer y/o escribir datos en la red.
java.applet	Contiene las clases necesarias para crear applets y se ejecutan en la ventana del navegador.
java.awt	Contiene clases para crear una aplicación GUI (Graphic User Interface) independiente de la plataforma.

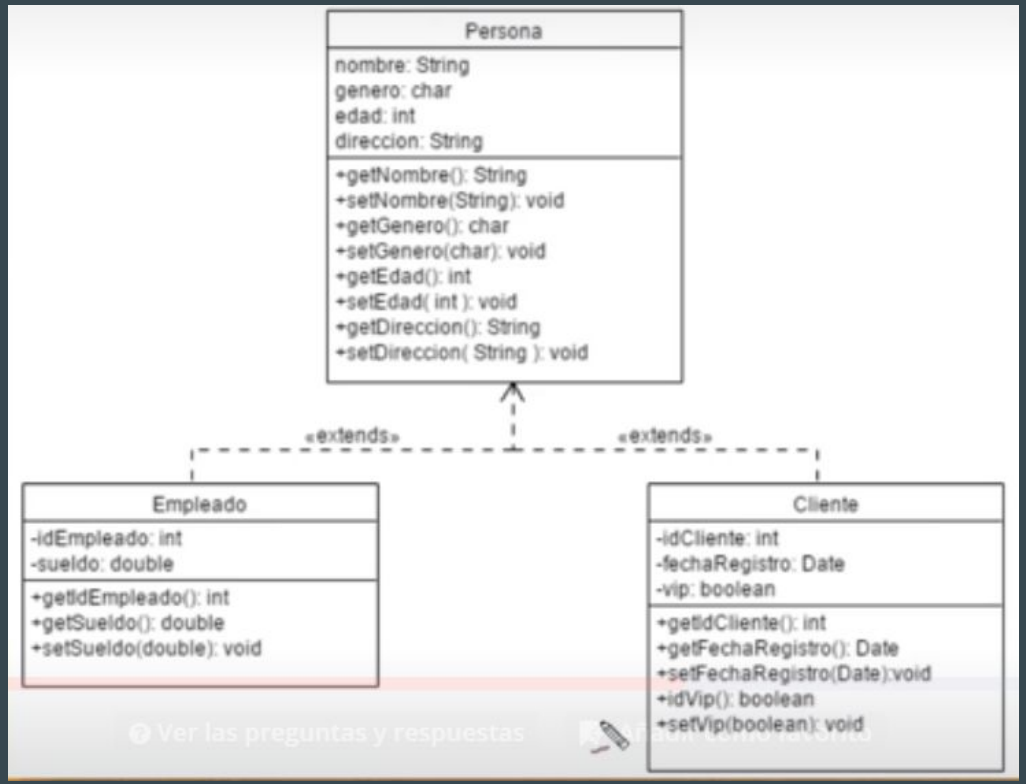
Ejemplo

Herencia

- Utilizar la estructura o modelo de una clase padre.
- La clase hija se puede acceder a los métodos y atributos de la clase padre.
- Se usa la palabra reservada “extends”, seguido del nombre de la clase de la cual se quiere heredar.
- Representar comportamiento común.
- Evitar duplicación de código.
- Es un característica de la POO.
- Se utiliza la palabra ”super” para acceder a los componentes del padre.



Ejemplo



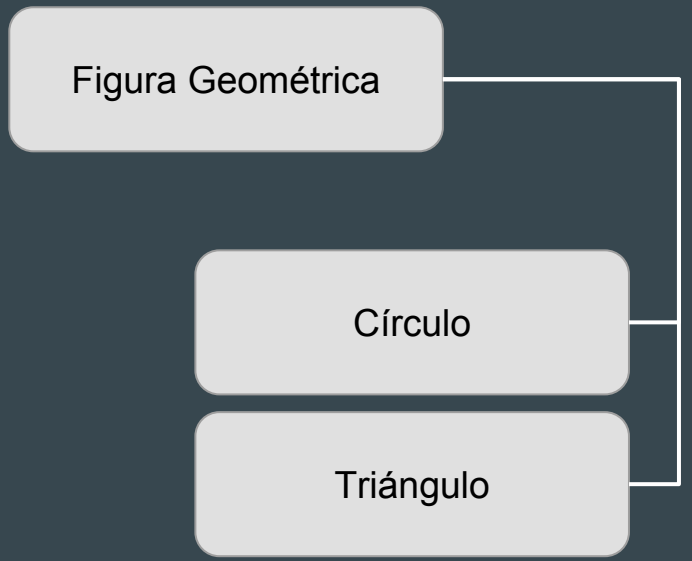
Clase Persona
+ getNombre()
+ setNombre()
+ getGenero(), etc

Clase Empleado
+getIdEmpleado()
+getSueldo(), etc

Polimorfismo

Es la capacidad que tienen los objetos de comportarse de múltiples formas. Eso significa que para aplicarlo se debe programar de manera general en vez de hacerlo de manera específica.

El objetivo de aplicar polimorfismo es el de poder crear un árbol de objetos en un proyecto, de tal manera que cada instancia de objeto que secree pueda pertenecer a un mismo tipo (siempre y cuando el análisis del desarrollo de los objetos lo necesite).



Ejemplo

Interfaces

- Declaración formal de un contrato.
- Son abstractos, es decir, no poseen implementación.
- Uso de la palabra reservada “implements”.
- Podemos implementar múltiples interfaces a una clase.

Estructura de una Interface

Definición de una interface en Java:

```
<modificadores>  interface <nombre_interface> [extends <interface padre>]
{
        <atributos>
        <métodos>
}
```

Uso de una interface en Java:

```
<modificadores> class <nombre_clase> [extends <superclase>] [implements
<interface1,interface2,etc>]
{
        <implementar_métodos_interface>
}
```

Ejemplo

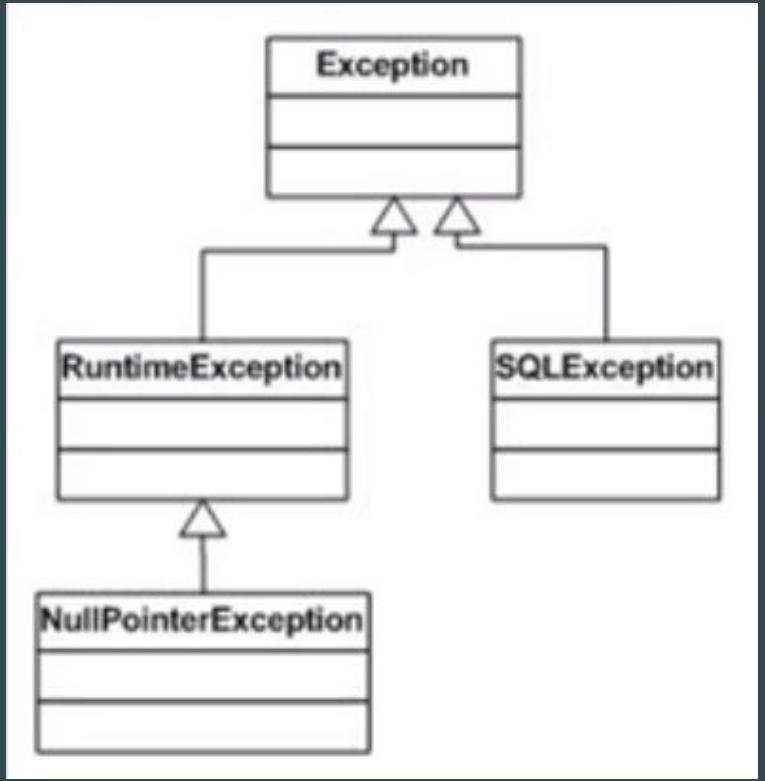
Excepciones

Check Exception

- Hereden de la clase Exception.
- Es obligatorio tratarlas.
- SQLException.

Unchecked Exception

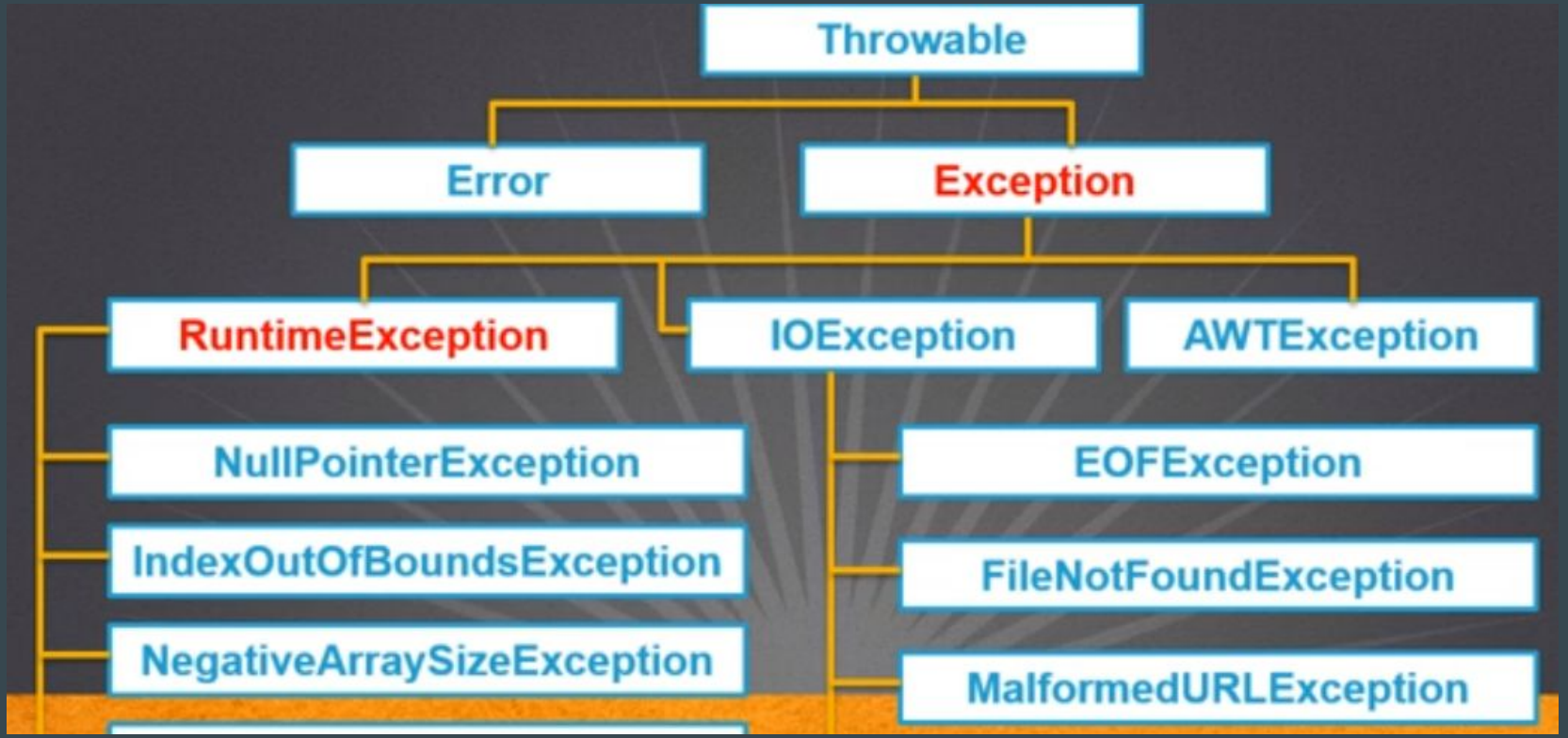
- Hereden de la clase Exception.
- Es opcional tratarla.
- RuntimeException.
    - NullPointerException.



Manejo de Excepciones

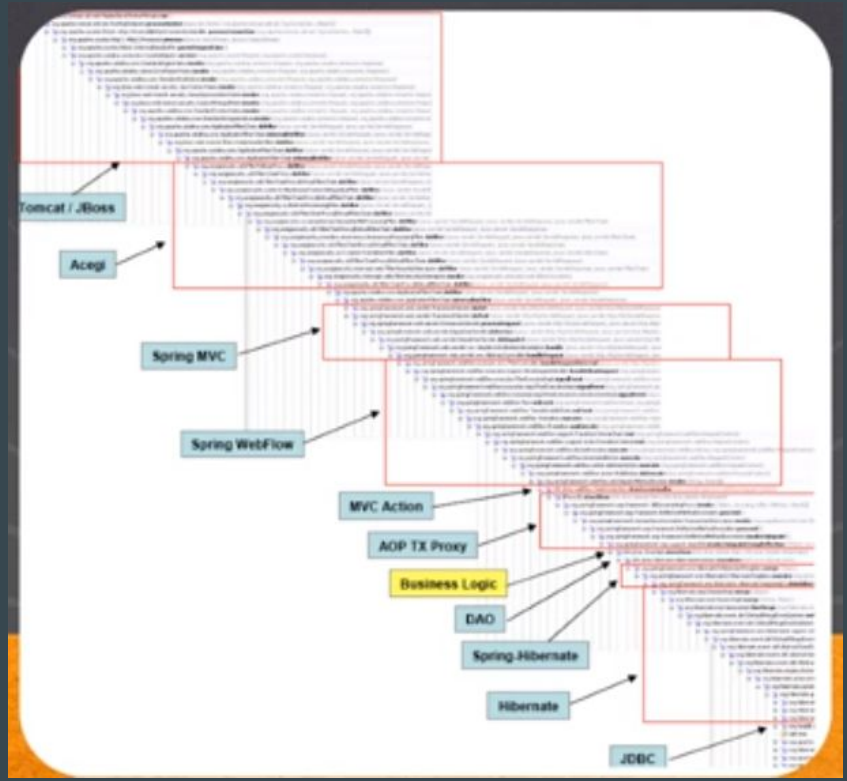
```
public void verificaExcepciones() {
   try {
      // código que lanza excepciones
   } catch (Exception ex) {
      //Bloque de código que maneja la excepción
      ex.printStackTrace();
   }
   finally{
      //Bloque de código opcional, pero
      //que se ejecuta siempre
   }
}
```

Excepciones más comunes



Stacktrace en Java

- Pilas de errores.
- Traza del error del inicio al fin.
- Flujo del error:
  - Método arroja un error.
  - Si no atrapa la excepción, la propaga hasta que alguna clase la maneja.
  - En caso de no manejarla el método main se satura
  - El programa se finaliza de manera anormal.



Cláusula Throws

```java
public class ArrojarExcepcion {

  public void metodoX() throws Exception {
    throw new Exception("Mensaje de error");
  }
}
```

```java
public class TestArrojarExcepcion {

  public static void main(String args[]) throws Exception {
    ArrojarExcepcion ae = new ArrojarExcepcion();
    ae.metodoX();
  }
}
```

Creación de Nuestras Excepciones

```
public class MiExcepcion extends Exception{

  public MiExcepcion(String mensaje){
    super(mensaje);
  }
}
```

```
public class ArrojarExcepcion2 {

  public void metodoX() throws MiExcepcion {
    throw new MiExcepcion("Mi mensaje de error");
  }
}
```

Ejemplo

Documentación con JavaDoc

Características:

- Es un paquete de Java.
- Genera automáticamente la documentación de nuestro programa.
- Funciona a través de etiquetas propias y de HTML.
- Crea un sitio web para nuestra documentación.
- Es una buena práctica realizar la documentación de nuestro programa.

Etiquetas para JavaDoc

Etiqueta	Descripción
@author	Nombre del autor del programa.
@deprecated	Indica que el elemento es obsoleto, pertenece a versiones anteriores y no se recomienda su uso.
@param	Definición de un parámetro de un método.
@return	Descripción de que devuelve el método, no se usa en constructores o métodos “void”.
@see	Indica que se asocia con otro método o clase.
@version	Versión del método o clase.
@excepcion	Nombre de la excepción más su descripción.
@throws	Nombre de la excepción más su descripción.

Ejemplo