



Programación en Java

POO (Encapsulamiento)

Encapsulamiento

- Sirve para:
 - Ocultar la Información
 - Protección de datos
 - Restringir el acceso a la información del Objeto (sus atributos).
- Se implementa a través de los modificadores de acceso.
- **Para encapsular un Objeto**, los **atributos** de la clase deben ser definidos como **privados** y se debe proporcionar acceso a cada uno de ellos a través de **métodos públicos (get/set – accesorios/mutadores)**.

Modificadores de acceso

- Determinan desde qué clases se puede acceder a un determinado elemento.
- **Si no especificamos** ningún modificador de acceso se utiliza el nivel de acceso por defecto o **default**.

Modificadores de acceso				
	La misma clase	Otra clase del mismo paquete	Subclase de otro paquete	Otra clase de otro paquete
public	X	X	X	X
protected	X	X	X	
default	X	X		
private	X			

Encapsulamiento en UML

- Las líneas describen las relaciones entre las Clases u Objetos (Asociación, Dependencia ó Generalización)
- Colocar el símbolo antes del nombre del atributo o método en el diagrama de clase.

SÍMBOLOS
" + " – Acceso Público (public)
" - " – Acceso Privado (private)
" # " – Acceso Protegido (protected)
" " – Cuando no hay símbolo significa que tiene acceso por Default (default)
OTRAS NOTACIONES
<i>Itálicas</i> – Para representar las Clases o Métodos Abstractos
<u>subrayado</u> – Para representar Métodos o Atributos Estáticos

Ejemplos de constructor

- Tanto los atributos como los métodos tienen modificador de acceso "**public**".
- Los objetos de la clase se crean dentro de la misma en el método main.

Persona

```
+nombre : String  
+direccion : String  
+telefono : String  
+edad : int  
-----  
+Persona()  
+caminar() : void  
+correr() : void  
+hablarPorTelefono() : void  
+main(entrada args[] : String) : void
```

Clase de prueba

Es recomendable **codificar la clase sin método main** y realizar la **creación de los objetos** deseados en una clase diferente ejecutable que puede llamarse "**PruebaNombreDeLaClase**" como en el ejemplo:

Persona

```
+nombre : String  
+direccion : String  
+telefono : String  
+edad : int  
-----  
+Persona()  
+caminar() : void  
+correr() : void  
+hablarPorTelefono() : void
```

PruebaPersona

```
-----  
+main(entrada args[] : String) : void
```



```
public class Persona {  
  
    //atributos  
    public String nombre;  
    public String direccion;  
    public String telefono;  
    public int edad;  
  
    //constructor  
    public Persona(){  
        System.out.println("Creando objeto con el constructor");  
    }  
  
    //metodos  
    public void caminar()  
    {  
        System.out.println("Persona camina");  
    }  
  
    public void correr()  
    {  
        System.out.println("Persona corre");  
    }  
  
    public void hablarPorTelefono()  
    {  
        System.out.println("Persona habla por telefono");  
    }  
  
}
```

Hecho por Huicho :)

Codificación de la clase Persona

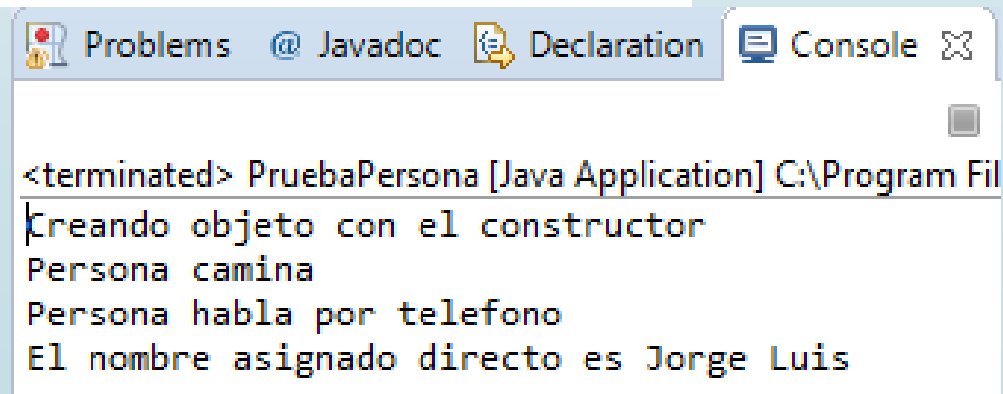
La clase se diseña y codifica de acuerdo a los atributos y métodos que la describan, **sin método main.**

Clase de prueba y salida en consola

```
public class PruebaPersona {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        //creacion de dos objetos de la clase Persona  
        Persona perso = new Persona();  
  
        //llamada a metodos  
        perso.caminar();  
        perso.hablarPorTelefono();  
  
        //modificando atributos directamente  
        perso.nombre="Jorge Luis";  
        System.out.println("El nombre asignado directo es " +  
            perso.nombre);  
    }  
}
```

La clase de **PruebaPersona** no describe una entidad o cosa por lo que **no tiene atributos ni métodos propios**, pero si el **método main** que la vuelve ejecutable.

Hecho por Huicho :)



```
Problems @ Javadoc Declaration Console  
  
<terminated> PruebaPersona [Java Application] C:\Program Fil  
|creando objeto con el constructor  
Persona camina  
Persona habla por telefono  
El nombre asignado directo es Jorge Luis
```


Encapsulamiento en UML

- Los **Atributos** por lo general serán **privados**, seguidos del Tipo de Dato y en algunos casos de una inicialización Explícita.
- Los **Métodos** por lo general serán **públicos**, seguidos de la lista de argumentos y del Tipo de Retorno.
- Los **Métodos** dentro del diagrama UML deben ser ordenados de la siguiente manera:

1. **Constructor**

2. **Métodos get/set**

3. **Métodos personalizados**

Ejemplo de clase con diferentes modificadores de acceso

Libro

```
-titulo : String
-autor : String
-editorial : String
-numPaginas : int = 200
-----
+Libro()
+getTitulo() : String
+setTitulo(nuevoTitulo : String) : void
+getAutor() : String
+setAutor(nuevoAutor : String) : void
+getEditorial() : String
+setEditorial(nuevaEditorial : String) : void
+getNumPaginas() : int
+setNumPaginas(paginas : int) : void
```

Se observan **métodos** que **no representan acciones** de la clase pero que sirven para **dar acceso a los atributos privados** (**set y get**)

Métodos set y get

- Si los atributos son **privados** solo son vistos por la **misma clase** por lo que no pueden ser asignados valores directamente desde una clase de prueba o se generará un error en tiempo de compilación indicando que no son visibles:

```
10 //tratando de modificar atributos
11 lib1.titulo= "Paciente cero";
12 lib1.autor= "Jonathan Maberry";
```

```
11 The field Libro.titulo is not visibleente cero";
```

- Para **acceder desde otra clase** como la de prueba a los valores de atributos privados se necesitan **métodos get/set**.

get ->

DEVUELVE el valor del atributo

NO recibe argumentos

Devuelve el tipo de dato del atributo

```
public String getTitulo() {  
    return titulo;  
}
```

set ->

ASIGNA valor al atributo

Recibe argumentos

Devuelve void

```
public void setTitulo(String titulo) {  
    this.titulo = titulo;  
}
```

```
public class Libro {  
  
    private String titulo;  
    private String autor;  
    private String editorial;  
    private int numPaginas= 200;  
  
    public Libro()  
    {  
        System.out.println("Se está creando objeto de Libro");  
    }  
  
    public String getTitulo() {  
        return titulo;  
    }  
    public void setTitulo(String titulo) {  
        this.titulo = titulo;  
    }  
    public String getAutor() {  
        return autor;  
    }  
    public void setAutor(String autor) {  
        this.autor = autor;  
    }  
    public String getEditorial() {  
        return editorial;  
    }  
    public void setEditorial(String editorial) {  
        this.editorial = editorial;  
    }  
    public int getNumPaginas() {  
        return numPaginas;  
    }  
    public void setNumPaginas(int numPaginas) {  
        this.numPaginas = numPaginas;  
    }  
}
```

Codificación de la clase Libro

En el contenido de los métodos **set()** es requerido el **operador this** para distinguir los atributos de las variables locales en la asignación y hacer referencia al objeto que los llama.

Clase de prueba y salida en consola

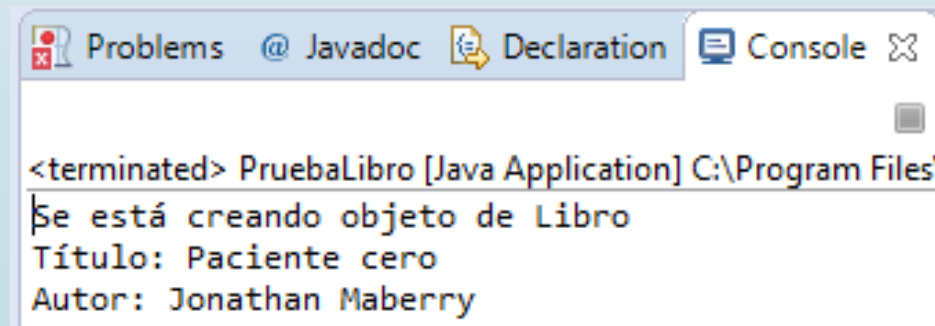
```
public class PruebaLibro {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        //creacion de objeto a partir de la clase  
        Libro lib1= new Libro();  
  
        //asignación de valores a atributos con objeto.set  
        lib1.setTitulo("Paciente cero");  
        lib1.setAutor("Jonathan Maberry");  
  
        //obtener valores de atributos con objeto.get  
        System.out.println("Título: " + lib1.getTitulo());  
        System.out.println("Autor: " + lib1.getAutor());  
    }  
}
```

La llamada a los métodos set y get se hace como con cualquier método:

objeto.método();

obj.getAtrib();

obj.setAtrib(valor);



The screenshot shows the 'Console' tab of a Java IDE. The output text is as follows:

```
<terminated> PruebaLibro [Java Application] C:\Program Files  
Se está creando objeto de Libro  
Título: Paciente cero  
Autor: Jonathan Maberry
```

Ejemplo de métodos set() y get()

Punto

-x : int

-y : int

+Punto()

+getX() : int

+setX(a : int) : void

+getY() : int

+setY(b : int) : void

```

/**
 * Clase que describe una entidad punto
 * @author Huicho
 */
public class Punto {

    /**
     * Atributos x,y de un punto de tipo float
     */
    private float x, y;

    /**
     * Constructor modificado para imprimir al crear objeto
     */
    public Punto(){
        System.out.println("Se está creando objeto de Punto");
    }

    /**
     * Método que devuelve el valor del atributo x
     * @return Devuelve x de tipo float
     */
    public float getX() {
        return x;
    }
}

```

PRIMERA PARTE DEL CÓDIGO

Hecho por Huicho :)

Codificación de la clase Punto

SEGUNDA PARTE DEL CÓDIGO

```

/**
 * Método que asigna valor al atributo x
 * @param x Recibe x de tipo float
 */
public void setX(float x) {
    this.x = x;
}

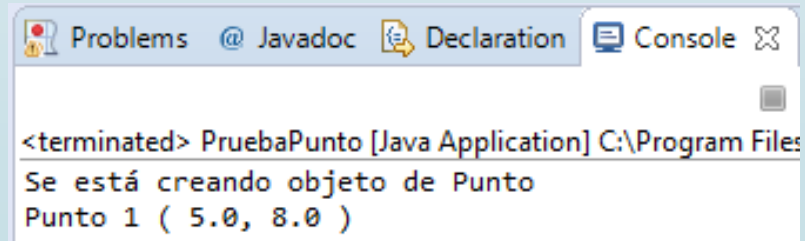
/**
 * Método que devuelve el valor del atributo y
 * @return Devuelve y de tipo float
 */
public float getY() {
    return y;
}

/**
 * Método que asigna valor al atributo y
 * @param y Recibe y de tipo float
 */
public void setY(float y) {
    this.y = y;
}
}

```


Clase de prueba y salida en consola

```
public class PruebaPunto {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        //creacion de objeto a partir de la clase  
        Punto punto1 = new Punto();  
  
        //modificando el atributo por su metodo set  
        punto1.setX(5);  
        punto1.setY(8);  
  
        System.out.println("Punto 1 ( " + punto1.getX() +  
            ", " + punto1.getY() + " )");  
    }  
}
```



The screenshot shows a console window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of the Java application. The text in the console is as follows:

```
<terminated> PruebaPunto [Java Application] C:\Program Files  
Se está creando objeto de Punto  
Punto 1 ( 5.0, 8.0 )
```

Documentación con javadoc

- Javadoc es una herramienta para generar documentación de código Java a partir de comentarios escritos con cierto formato que se escriben en el código fuente.
- El archivo javadoc.exe forma parte de la instalación del jdk y se ubica en la carpeta bin junto con otros binarios como el compilador javac.exe.
- La documentación se exporta en formato HTML por lo cual se necesita un navegador web para visualizarla.

Documentación con javadoc

- Escribir comentarios con los delimitadores sobre cualquier elemento como atributos, métodos y constructores:

```
/**  
 * Comentarios para documentación  
 */
```

- Comentario general y autoría al inicio de la clase

```
/**  
 * Descripción general de la clase o elemento  
 * Su declaración inicia justo después de comentario  
 * @author Huicho  
 */
```

Javadoc en línea de comandos

javadoc [opciones] Clase

Opciones:

javadoc -public Clase

Muestra sólo las clases y los miembros públicos.

javadoc -protected Clase

Muestra clases y miembros protegidos (protected) y públicos.

javadoc -package Clase

Muestra clases y miembros con acceso de paquete, protegidos y públicos.

javadoc -private Clase

Muestra todas las clases y todos los miembros, incluyendo los privados.

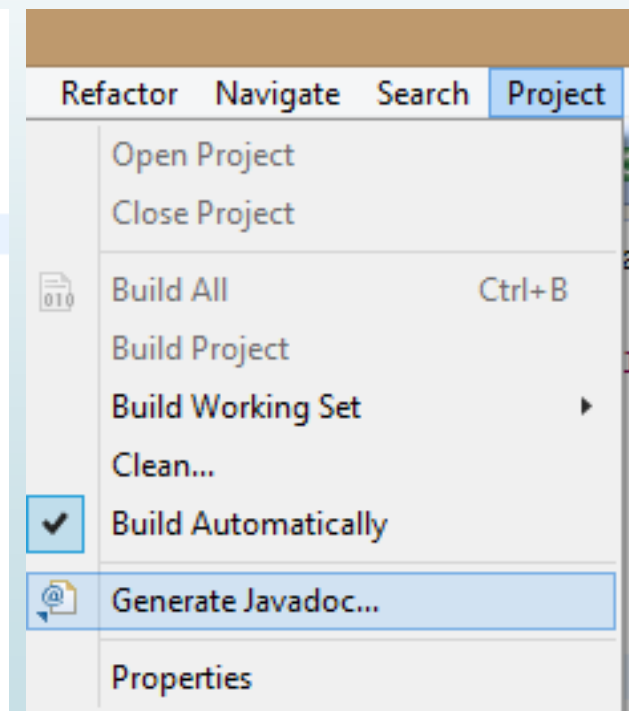
Javadoc en Eclipse

► Menú Project -> **Generate Javadoc...**

```
/**
 * Clase que describe una entidad punto
 * @author Huicho
 */
public class Punto {

    /**
     * Atributos x,y de un punto de tipo float
     */
    private float x, y;

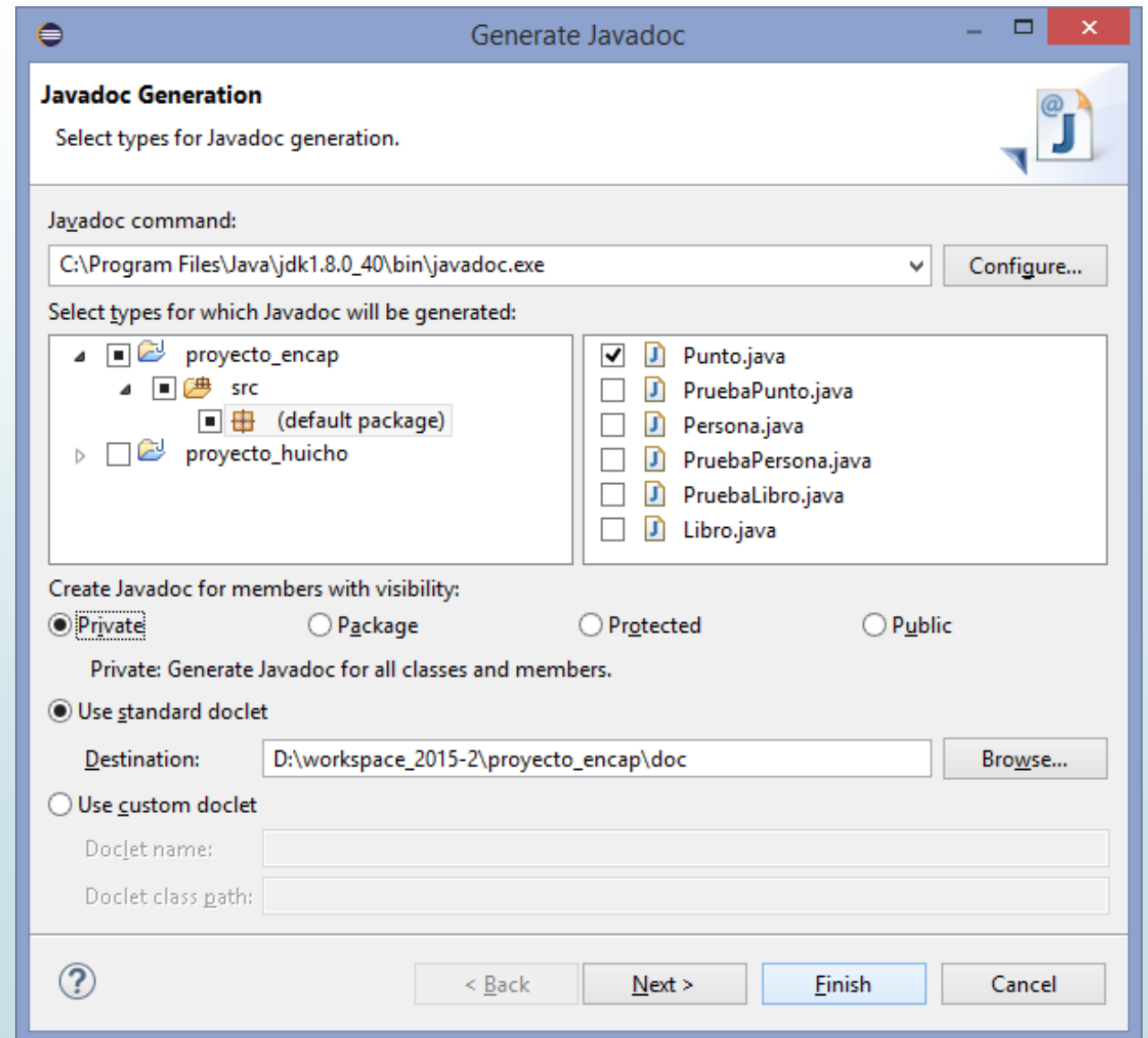
    /**
     * Constructor modificado para imprimir al crear objeto
     */
    public Punto(){
        System.out.println("Se está creando objeto de Punto");
    }
}
```



Hecho por Huicho :)

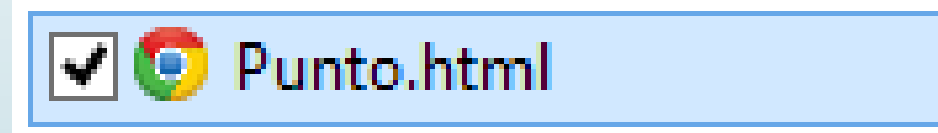
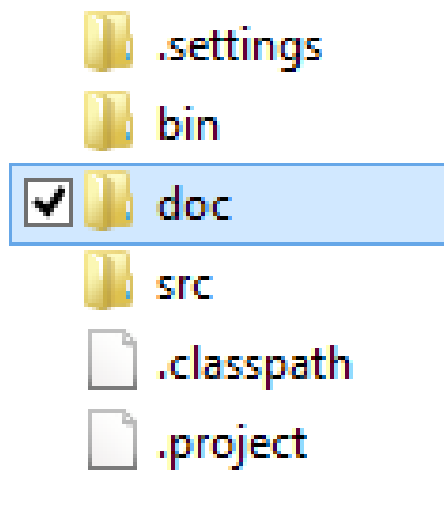
- Click **Configure...** en buscar "javadoc.exe" en ruta de jdk.
- Seleccionar clase a documentar.
- Marcar con visibilidad **Private**.
- Indicar ruta de destino.

Hecho por Huicho :)



Generar documentación en Eclipse

- Ubicar en la carpeta del proyecto la recién creada **carpeta "doc"** y acceder a ella.
- Javadoc genera un **archivo con extensión .html** con el nombre de la clase seleccionada.



PACKAGE **CLASS** USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Punto

java.lang.Object
Punto

```
public class Punto  
extends java.lang.Object
```

Clase que describe una entidad punto

Author:

Huicho

Field Summary

Fields

Modifier and Type	Field and Description
private float	x Atributos x,y de un punto de tipo float
private float	y Atributos x,y de un punto de tipo float

Se muestran los comentarios anotados sobre la definición de la clase, del autor y un resumen de los atributos.

Constructor Summary

Constructors

Constructor and Description

Punto()

Constructor modificado para imprimir al crear objeto

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type

Method and Description

float

getX()

Método que devuelve el valor del atributo x

float

getY()

Método que devuelve el valor del atributo y

void

setX(float x)

Método que asigna valor al atributo x

void

setY(float y)

Método que asigna valor al atributo y

El comentario sobre el constructor es mostrado y un resumen de los métodos contenidos en la clase con su descripción.

Method Detail

getX

```
public float getX()
```

Método que devuelve el valor del atributo x

Returns:

Devuelve x de tipo float

setX

```
public void setX(float x)
```

Método que asigna valor al atributo x

Parameters:

x - Recibe x de tipo float

Después de muestran a detalle la definición y descripción de cada elemento comentado como el constructor, los atributos y los métodos.