



Fecha: 2 Semana

Tema: Inyección de dependencias

Inyección de dependencias

```
E 💲 👂 🕴 1 package com.curso;
> 🚰 > HolaWord [AcademiaJava main]

✓ 

✓ InyeccionDepedsDI

                                        public class Juguete {
 > M JRE System Library [JavaSE-1.8]
                                              private String producto;
     > # com.curso
                                              Carrito Matel;

y 

→ com.curso.constructor

      >   Carrito.java
                                              public Juguete(String producto) {
      > 🔝 Inyector.java
      > Duguete.java
> Pilas.java
                                                             this.producto = producto:
      > 🗓 Principal.java
                                              void baterias() {
       > II Tractor.java
                                                   System.out.print(producto);
                                    13

→ 

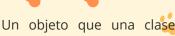
⊕ com.curso.setter

                                    14
                                                   Matel = new Carrito("Hot Wheels");
       Carrito.java
                                    15
                                                   Matel.baterias();
      > Inyector.java
                                    16
        Juguete,java
                                    17 }
        Pilas.java
      > D Principal.iava
       → I Tractor.java

→ ⊕ com.curso.variable

      > D Carrito.iava
      > 🗓 Inyector.java
      > / Juguete.java
> / Pilas.java
      >  Principal.iava
```

Conceptos Clave



- **Dependencia**: Un objeto que una clase necesita para funcionar correctamente.
- Inyección de Dependencias: El proceso mediante el cual un contenedor o framework proporciona las dependencias necesarias a un objeto.
- Inversión de Control: Un principio de diseño donde el control del flujo del programa se invierte, delegando la responsabilidad de crear y gestionar las dependencias a un contenedor externo.

1. Vamos a aplicar la Inyección de Dependencias

Interfaz Pilas: Primero, crearemos una interfaz para Pilas para permitir múltiples implementaciones.

```
Carrito.java

Juguete.java
Principal.java
Tractor.java

package com.curso.setter;

public interface Pilas {
 public void baterias();
}

7
```

Implementación de Pilas: Implementaremos la interfaz Pilas.

```
package com.curso.variable;
 1
  3
    public class Carrito implements Pilas{
 4
  5
        private String version;
  6
  7Θ
        public Carrito(String version) {
 8
            this.version = version;
 9
10⊖
        @Override
△11
        public void baterias() {
        System.out.println( " Con baterias incluidas: " + version);
12
13
14 }
15
```

Clase Juguete: Modificaremos Juguete para recibir una instancia de Carrito a través de su constructor.





Fecha: 2 Semana

Tema: Inyección de dependencias

```
Pilas.java

    Juguete.java 

    X

   package com.curso.variable;
 2
 3
   public class Juguete {
 4
 5
       private String producto;
 6
       private Carrito mattel;
 7
 80
       public Juguete(String producto) {
 9
                    this.producto = producto;
10
11⊖
       public Carrito getMattel() {
12
            return mattel;
13
       public void setMattel(Carrito mattel) {
14⊖
            this.mattel = mattel;
15
16
17⊖
       void baterias() {
18
            System.out.print(producto);
19
            mattel.baterias();
20
        }
21 }
22
```

Clase Principal: Crearemos una clase principal para probar nuestra inyección de dependencias.

```
Carrito.java
            Pilas.java
                       Juguete.java
                                    package com.curso.variable;
 2
 3
    public class Principal {
 4
 5⊝
        public static void main(String[] args) {
 6
 7
            Juguete juguete = new Juguete("Hot-Whales Mx");
 8
 9
            Invector.invectarCarrito(juguete);
10
11
            juguete.baterias();
12
13 }
14
```

Explicación

- 1. Interfaz Pila: Define el contrato que cualquier implementación de carrito debe cumplir.
- 2. Implementación Pilas: Proporciona una implementación concreta de la interfaz Pilas.
- 3. Clase Juguete: Recibe una instancia de Carrito a través del constructor, lo que permite la inyección de dependencias.
- 4. Clase Principal: Crea una instancia de Carrito y la inyecta en Juguete al crear su instancia.

