



## Inyección de dependencias

```
1 package com.curso;
2
3 public class Jugete {
4
5     private String producto;
6     Carrito Matel;
7
8     public Jugete(String producto) {
9
10         this.producto = producto;
11     }
12
13     void baterias() {
14         System.out.print(producto);
15         Matel = new Carrito("Hot Wheels");
16         Matel.baterias();
17     }
18 }
```

### 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.

```
1 package com.curso.setter;
2
3 public interface Pilas {
4     public void baterias();
5 }
6
7 }
```

Implementación de Pilas: Implementaremos la interfaz Pilas.

```
1 package com.curso.variable;
2
3 public class Carrito implements Pilas{
4
5     private String version;
6
7     public Carrito(String version) {
8         this.version = version;
9     }
10
11     @Override
12     public void baterias() {
13         System.out.println( " Con baterias incluidas:" + version);
14     }
15 }
```

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





```
Carrito.java  Pilas.java  Juguete.java X
1 package com.curso.variable;
2
3 public class Juguete {
4
5     private String producto;
6     private Carrito mattel;
7
8     public Juguete(String producto) {
9         this.producto = producto;
10    }
11    public Carrito getMattel() {
12        return mattel;
13    }
14    public void setMattel(Carrito mattel) {
15        this.mattel = mattel;
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  Principal.java X
1 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         Inyector.inyectarCarrito(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.

