Natalia Denise Fuentes Rubio
Semana 2 – Evidencias
Academia Java

Tarea 3: Realizar un programa documentado donde se utilice la inyección de dependencias y explicar el código

La inyección de dependencias es una práctica de programación donde se implementa un patrón de diseño orientado a objetos, en el que un objecto o función recibe sus dependencias de una fuente externa en lugar de crearlas. Esto significa que los objetos cumplen aspectos que necesitan de otras clases para poder funcionar; las clases no crean los objetos que necesitan, sino que son suministrados desde otra clase que inyectará la implementación deseada.

Hay 3 formas de inyectar dependencias en las clases; por variable, por setters y por constructores.

En este caso se mostrará la inyección de dependencias mediante variables; simularemos un programa donde se involucra un adoptante y un gato que será adoptado.

```
package com.inyecciondep;

//Se decalra la clase del Adoptante
public class Adoptante {

	//Se definen sus atributos
	private String nombre;
	Gato gatoAdoptado;

	//Se genera el constructor del adoptante
	public Adoptante(String nombre) {
		this.nombre = nombre;
	}


	//Se crea un método para adoptar un gato
	void adoptarGato() {
		System.out.println(nombre);
		gatoAdoptado.adoptar();
	}



}
```

Se tiene la clase pública adoptante en donde se define el atributo *nombre* de tipo privado, y el atributo *gatoAdoptado* del tipo Gato que almacenará que el gato ha sido adoptado.

En el constructor se toma el *nombre* y se le asigna al atributo *nombre* del *Adoptante*

Se prosigue con el método *adoptarGato* que no toma ningún parámetro y no devuelve nada.

Se declara la interfaz gato en donde se define el método *adoptar*.

```
package com.inyecciondep;

//Se declara interface Gato
```

```java
public interface Gato {

	//Se define método adoptar
	public void adoptar();

}
```

Posteriormente se declaran las clases *GatoTuxedo* y *Gato Naranja*, en donde ambas implementan la interfaz Gato, dentro de cada una se define el atributo color de forma privada, se declaran sus constructores respectivamente y se implementa el método adoptar.

```java
package com.inyecciondep;

//Se declara la clase GatoTuxedo que implementa a Gato
public class GatoTuxedo implements Gato{

	//Se define el atributo color de forma privada
	private String color;

	//Se declara el constructor del GAtoTuxedo
	public GatoTuxedo(String color) {
		this.color = color;
	}

	//Se implementa el método adoptar
	@Override
	public void adoptar() {
		System.out.println("Adoptaste al gato " + color);
	}

}
```

```java
package com.inyecciondep;

public class GatoNaranja implements Gato{

	private String color;

	public GatoNaranja(String color) {
		this.color = color;
	}

	@Override
	public void adoptar() {
		System.out.println("Adoptaste al gato " + color);
	}

}
```

Se prosigue con la creación de la clase *Inyector* en donde se declara el método *inyectarGato* en donde se toma el objeto *Adoptante* y se le inyecta un gato de alguna de las clase *GatoNaranja* o *GatoTuxedo*.

```
package com.inyecciondep;

//Se declara la clase inyector
public class Inyector {

	//Se declara método estático
	static void inyectarGato(Adoptante adop) {
		adop.gatoAdoptado = new GatoNaranja("naranjoso");
	}

}
```

Como parte final, se crea la clase *Principal* en donde se crea una instancia con el nombre del *adoptante Karen*, se implementa el método *inyectarGato* para asignarle un gato, en este caso el naranja, para que al final se muestre el nombre de que *Karen adoptó al gato naranja.*

```
public class Principal {

	public static void main(String[] args) {

		Adoptante adoptante = new Adoptante("Karen: ");

		Inyector.inyectarGato(adoptante);

		adoptante.adoptarGato();
	}

}
```

De igual forma, se modificó el programa para que presente inyección mediante setters, en este caso solo se modificaron las clases *Adoptante* y el *Inyector.*

En la clase del *adoptante*, se definieron los métodos get y set que permiten inyectar al *Gato*.

```
package com.inyecciondep.setter;

public class Adoptante {

	private String nombre;
	private Gato gatoAdoptado;

	public Adoptante(String nombre) {
		this.nombre = nombre;
	}
```

```
	//Se definieron los getters y setters
	public Gato getGatoAdoptado() {
		return gatoAdoptado;
	}

	public void setGatoAdoptado(Gato gatoAdoptado) {
		this.gatoAdoptado = gatoAdoptado;
	}

	void adoptarGato() {
		System.out.println(nombre);
		gatoAdoptado.adoptar();
	}



}
```

En la clase del *Inyector*, se redefinió el método estático implementando un método setter, con el cuál se inyectara al *adoptante* con un gato.

```
package com.inyecciondep.setter;

public class Inyector {

	static void inyectarGato(Adoptante adop) {
		adop.setGatoAdoptado(new GatoNaranja("naranjoso"));
	}

}
```

Como último caso de inyección de dependencias, se tiene la inyección mediante constructores. Para este caso, se modificaron de nuevo las clases *Adoptante* e *Inyector.*

En la clase *Adoptante* se define el constructor de la clase, en donde se recibe el nombre del *adoptante* como el *gatoAdoptado*.

```
package com.inyecciondep.constructor;

public class Adoptante {

	private String nombre;
	private Gato gatoAdoptado;

	//Se define el constructor
	public Adoptante(String nombre, Gato gatoAdoptado) {
		this.nombre = nombre;
		this.gatoAdoptado = gatoAdoptado;
```

```
	}


	void adoptarGato() {
		System.out.println(nombre);
		gatoAdoptado.adoptar();
	}



}
```

Por su parte, en la clase *Inyector,* el método crea las instancias para ambos gatos paraluego devolver una nueva instancia del *adoptante* con su nuevo gato adoptado.

```
package com.inyecciondep.constructor;

public class Inyector {

	static Adoptante getAdoptante() {
		Gato GatoNaranja = new GatoNaranja("naranjoso");
		Gato GatoTuxedo = new GatoTuxedo("tuxedo");
		return new Adoptante("Raúl", GatoTuxedo);
	}

}
```