Ejercicio Código 5

Autor: José Guillermo Sandoval Huerta

Fecha: 09 noviembre 2023

Inyección de Dependencias

La inyección por dependencia, se refiere a un objeto que otra clase necesita para realizar su trabajo. Esto implica suministrar las dependencias de una clase desde el exterior, en lugar de que la clase misma las cree. Se tiene tres tipos de inyecciones de dependencias:

- Inyección a través de variable.
- Inyección a través de setter.
- inyección a través de constructor.

Y partiendo de lo anterior se diagramo lo siguiente:

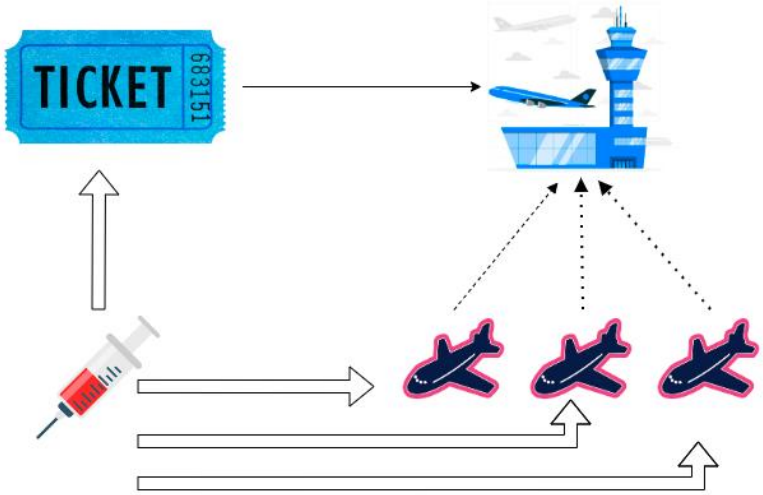

Imagen 1. Representación gráfica del programa realizado.

Donde se implementaron las siguientes clases:

Código 1. Interfaz Avión.

```
…

public interface Avion {
        public void reservacion();
}
```

Código 2. Clase Volaris que implementa la interfaz Avión.

```
…

public class Volaris  implements Avion {
        private String modelo;

        public Volaris(String  modelo) {
                this.modelo  = modelo;
        }

        @Override
        public void reservacion()  {
                System.out.println("Avion:  Volaris - " + modelo);
        }

}
```

Se crearon la clase Aeroméxico y VivaAerobús con la misma estructura de Volaris.

Código 3. Clase Ticket.

```
…

public  class Ticket {
        private String numeroTicket;
        private String nombrePasajero;
        private String destino;
        private String fechaSalida;
        Avion avion;
        private Avion avion2;

        public  Ticket(String numeroTicket, String nombrePasajero, String destino, String fechaSalida) {
                this.numeroTicket = numeroTicket;
                this.nombrePasajero = nombrePasajero;
                this.destino = destino;
                this.fechaSalida = fechaSalida;
        }

        public  void setAvion2(Avion avion2) {
                this.avion2 = avion2;
        }

        void reservarAvion() {
                System.out.println("**************************" +
                                                "\nNúmero Ticket: " + numeroTicket +
                                                "\nPasajero: " + nombrePasajero +
                                                "\nDestino: " + destino +
                                                "\\nFecha Salida: " + fechaSalida);
                avion.reservacion();
        }

        void reservarAvion2() {
                System.out.println("**************************" +
                                                "\nNúmero Ticket: " + numeroTicket +
                                                "\nPasajero: " + nombrePasajero +
                                                "\nDestino: " + destino +
                                                "\\nFecha Salida: " + fechaSalida);
                avion2.reservacion();
        }

}
```

Donde declara la estructura para los tres diferentes tipos de instancias cada uno con una inyección de dependencia diferente.

Código 4. Clase InyectorAvion.

```
…

public class InyectorAvion {

	static Volaris volaris = new Volaris("AirbusA320");
	static Aeromexico aeromexico = new Aeromexico("Boeing 787");
	static VivaAerobus vivaAerobus = new VivaAerobus("Airbus A320neo");

	static void inyectarAvion(Ticket ticket, String avion) {
		if(avion == "vs")
			ticket.avion = volaris;
		else if(avion == "am")
			ticket.avion = aeromexico;
		else if(avion == "va")
			ticket.avion = vivaAerobus;
	}

	static void inyectarAvion2(Ticket ticket, String avion) {
		if(avion == "vs")
			ticket.setAvion2(volaris);
		else if(avion == "am")
			ticket.setAvion2(aeromexico);
		else if(avion == "va")
			ticket.setAvion2(vivaAerobus);
	}

	static Ticket2 inyectarAvion3(String avion) {
		if(avion == "vs")
			return new Ticket2("A106", "Patrobas", "Cancún", "Enero 2024", volaris);
		else if(avion == "am")
			return new Ticket2("JA20", "Epeneto", "Monterrey", "Abril 2024", aeromexico);
		else if(avion == "va")
			return new Ticket2("C002", "Andronico", "Guanajuato", "Diciembre 2023", vivaAerobus);

		return null;
	}
}
```

Esta clase es la que permite la inyección por dependencias.

Código 5. Clase Principal.

```
…

public class Principal {
	public static void main(String[] args) {

		//Inyeccion a traves de variable
		Ticket vuelo1 = new Ticket("A106", "Patrobas", "Cancún", "Enero 2024");
		InyectorAvion.inyectarAvion(vuelo1, "vs");
		vuelo1.reservarAvion();

		//Inyeccion a traves de setter
		Ticket vuelo2 = new Ticket("JA20", "Epeneto", "Monterrey", "Abril 2024");
		InyectorAvion.inyectarAvion2(vuelo2, "am");
		vuelo2.reservarAvion2();

		//Inyeccion a traves de constructor
		Ticket2 vuelo3 = InyectorAvion.inyectarAvion3("va");
		vuelo3.reservarAvion();
	}
}
```

En esta clase se instancias los objetos y se inyectan su respectiva dependencia.