Factory Pattern

Federico Degan

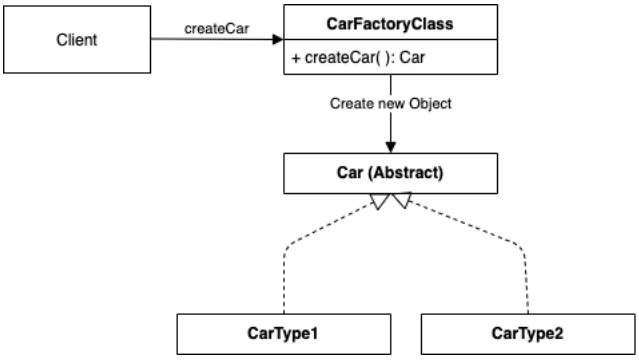
Kurzbeschreibung

Die Factory Method ist ein Erzeugungsmuster, anstatt Objekte direkt über einen Konstruktor zu erstellen übernimmt eine externe Klasse über Methoden das Erstellen von Objekten.

Zweck

Der Vorteil des Factory Patterns ist, dass man nicht eine konkrete Implementierung der Objektklassen macht, so lässt sich ein Projekt sehr einfach um neue Objekttypen erweitern.

UML-Diagramm



Beschreibung

Bei der Factory Method bzw. Pattern handelt es sich um ein Erzeugungsmuster, heisst es definiert wie ein Objekt erstellt wird.
Die definierende Charakteristik des Factory Patterns ist, dass die Objekte über das Aufrufen einer Methode erstellt werden und nicht wie sonst über einen Konstruktor.
Dazu erstellt man eine Factoryklasse die für das Handling der Erstellung von neuen Objekten verantwortlich ist und einem über eine Methode neue Objekte liefert, die Objekte werden von einer abstrakten Klasse abgeleitet..
Das heisst die Erstellung von neuen Objekten wird in Unterklassen verlagert, was die zu erstellenden Objekten nicht von einer konkreten Implementierung abhängig machen.
Das Factory Pattern wird grundsätzlich immer mit einem Interface verwendet um die Einheitliche Implementierung der verschiedenen Objektklassen zu garantieren.

Beispiele

Die Factory Methode kann so einfach sein wie eine Statische Methode die ein Objekt erzeugt, so ruft man anstelle von

```
Object object = New Object( );
```

Einfach eine dafür vorgesehene Methode auf, deren return Wert ein neues Objekt ist

```
Object object = objectFactory.createNewObject( );
```

Beispielcode in Java

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

		// Ich suche nach IRGENDEINEM Auto für den Alltag
		Car auto = CarFactory.findCar("daily");
	    auto.details();

	    // Ich suche nach einem Nutzfahrzeug
	    auto = CarFactory.findCar("utility");
	    auto.details();

	    // Ich suche nach einem sportlichen Fahrzeug
	    auto = CarFactory.findCar("sporty");
	    auto.details();
	}
}

class CarFactory
{
  public static Car findCar(String use)
  {
    if ( use.equals("daily") )
      return new BMW();

    else if ( use.equals("sporty") )
      return new Porsche();

    else if ( use.equals("utility") )
      return new Ford();

    return null;
  }
}

interface Car
{
  public void details();
}

public class BMW implements Car
{
	public void details() {
		System.out.println("BMW den du zur Abeit fährst");
	}
}

public class Ford implements Car
{
	public void details() {
		System.out.println("Ford F150 mit dem du Dinge transportierst");
	}
}

public class Porsche implements Car
{
	public void details() {
		System.out.println("Porsche für Sonntagsfahrt");
	}
}
```