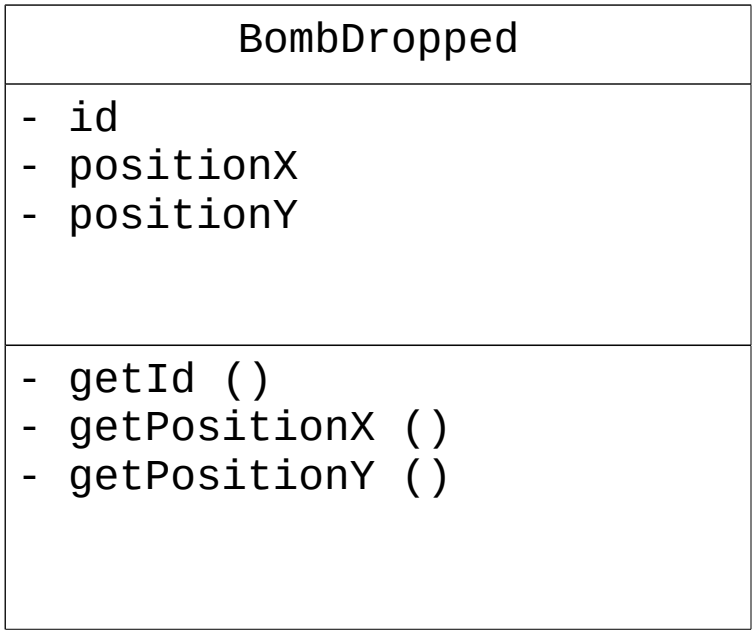
Yanik Künzi

Value Object Pattern

Einsatzzweck

Das Value Object Pattern, auch Transfer Object Pattern genannt, wird eingesetzt, wenn mehrere Informationen innerhalb eines Objekts gesendet werden müssen. In unserem Fall, dem Protokoll, wird das Design Pattern verwendet, um die Messages zu implementieren. Die Implementierung des Value Object Patterns ist sehr einfach. Zur Vorstellung des Design Patterns habe ich die Klasse BombDropped gewählt. Diese hat drei Membervariablen. Die eine ist die *id*, worüber die Bombe eindeutig identifiziert werden kann. Des weiteren gibt es noch die *positionX* und die *positionY*, welche die X-, bzw. Y-Koordinate angeben, an der die Bombe platziert wurde. Der Datentyp aller Variablen ist int und die Variablen können über eine entsprechende get-Funktion abgerufen werden. In einem Klassendiagramm sieht das folgendermassen aus:



Yanik Künzi

Beschreibung

Die Objekte der BombDropped Klasse sind simple POJO (Plain Old Java Object) Objekte. Es wird jedoch keine Logik in die Klasse implementiert. Die einzigen Methoden, in einer Klasse, welche das Value Object Pattern einsetzt, sind getter und setter. In unserem Fall wurde auf setter verzichtet, da wir noch ein weiteres Design Pattern (Immutable Object) angewandt haben, wodurch das verwenden von Setter-Methoden hinfällig wurde. Das Objekt enthält also nur Daten und hat kein besonderes Verhalten implementiert. Des weiteren muss das Objekt der Klasse serialisiert werden können, um es im Netzwerk an einen Client oder Server senden zu können. Wie bereits aus dem Klassendiagramm auszulesen ist, ist das Value Object Pattern ein sehr einfach zu implementierendes Entwurfsmuster, ist aber bei der Kommunikation über das Netzwerk sehr hilfreich und wird deshalb in diesem Bereich häufig eingesetzt.

Probleme bei einer Lösung ohne Value Object

Wäre bei der Implementierung des Netzwerk & Protokolls auf den Einsatz des Value Object Entwurfsmuster verzichtet worden, wären verschiedenste Probleme und Unschönheiten aufgetreten. Das grösste Problem wäre sicherlich die Übertragung der Daten geworden. Da alle Variablen einzeln hätten übertragen werden müssen, wäre der Netzwerk Teil sehr statisch geworden, was grösseren Wartungsaufwand nach sich gezogen hätte. Des weiteren wären Erweiterung sehr mühsam gewesen. Durch den Einsatz des Value Object Patterns, werden alle zu einer Message gehörenden Variablen an einem Ort gespeichert, was den Zugriff auf diese stark vereinfacht. Dadurch wird zudem Write-only oder Spaghetti-Code vermieden.

Yanik Künzi

Anwendung auf eigenen Code

```
 1 package protocol;
 2
 3 import network.Message;
 4
 5 final public class BombDropped implements Message {
 6     final private int id, positionX, positionY;
 7
 8     public BombDropped (int id, int positionX, int positionY)
 9     {
10         this.id = id;
11         this.positionX = positionX;
12         this.positionY = positionY;
13     }
14
15     public int getId() {
16         return id;
17     }
18
19     public int getPositionX() {
20         return positionX;
21     }
22
23     public int getPositionY() {
24         return positionY;
25     }
26 }
~
```

Hier zu sehen ist die DropBomb Klasse, welche eine Kindklasse von Message ist. Sie besitzt drei Eigenschaften. Diese sind *id* welche die Bombe eindeutig identifiziert und *positionX* und *positionY* welche den X und Y Wert auf dem Koordinatensystem angibt, um anzuzeigen, wo die Bombe platziert wurde. Für alle Variablen sind Getter-Methoden vorhanden, welche die Variablen zurückgeben. Objekte diese Klassen werden wie oben beschrieben serialisiert und dann über das Netzwerk dem Server geschickt. Wie zu sehen ist, enthält die Klasse keinerlei Logik, da es nur für die Übertragung von Variablen über das Netzwerk eingesetzt wird.