Objektorientierte Programmierung WS 2013/2014 - Aufgabenblatt 6

Prof. Dr. U. Kastens
Institut für Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn

Ausgabe: 07.01.2014

Aufgabe 1 (Das Entwurfsmuster "Oberserver")

Das Beobachter-Muster (siehe Folie 307) ist eines der bekanntesten und meistverwendeten Entwurfsmuster. Deshalb hat es auch Eingang in die Java-Standardbibliothek gefunden. Dort existiert das Interface `java.util.Observer` und die Klasse `java.util.Observable`.

Das Verzeichnis `blatt6/observer` enthält eine Beispiel-Anwendung dieses Musters.

a) Ordnen Sie die Rollen des Oberserver-Musters den Klassen des Beispiels zu.
b) Die Tatsache, dass `Observable` eine Klasse ist, verhindert es zusammen mit der Einfacherbung in Java, dass `Ding` von einer anderen Klasse, z.B. der Klasse `UeberDing`, erben kann.

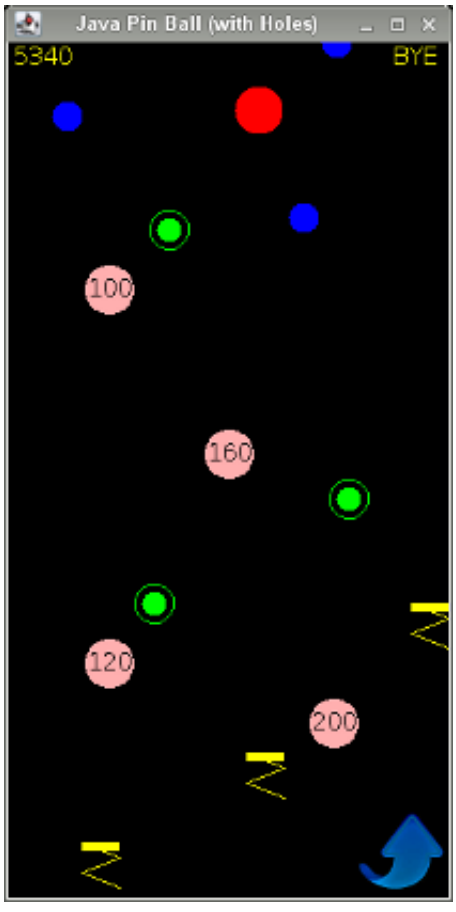
   Ändern Sie die Klasse `Ding` so ab, dass sie von `UeberDing` erbt und die `Observable`-Funktionalität trotzdem bereitstellt.

   `UeberDing` könnte z.B. so aussehen:

```
class UeberDing
{ // unnuetze Oberklasse von Ding
}
```

Aufgabe 2 (Inkrementelle Weiterentwicklung)

Das Verzeichnis `blatt6/pinballgame` enthält die Java-Implementierung einer Flipper-Simulation:



Unser Flipperspiel soll nun um "Löcher" erweitert werden. Trifft eine Kugel auf ein Loch, so verschwindet sie aus dem Spiel. Ein Loch soll als roter Kreis dargestellt werden.

Folie 220 behandelt die Möglichkeit, ein solches Loch durch inkrementelle Weiterentwicklung aus der Klasse `Ball` zu implementieren. Dazu erbt die Klasse `Loch` von `Ball`, obwohl sie konzeptionell kein Untertyp von `Ball` ist.

Eine zweite Möglichkeit, die Klasse `Loch` zu realisieren, ist die Verwendung der Komposition (Folie 222), wobei Aufgaben an ein `Ball`-Objekt delegiert werden.

Erweitern Sie das beiliegende Flipperspiel um Löcher. Implementieren Sie dabei beide oben beschriebenen alternativen Modelle.