

Übungsblatt 8

Aufgabenlösung

Abgabe: 16.12.2012

Aufgabe 1 Alles super hier (20%)

In unserem Spiel sind 2 Klassen implementiert dessen Instanzen aufgehoben werden können: Hammer und Scissor. Sie besaßen schon die Superklasse `Collectable` seit einigen Abgaben. Auch die Klassen der Hindernisse: Bush, Stone und Comb erbten schon von der Superklasse `Obstacle`. Da wir aber in vorherigen Aufgaben mit Hilfe vom *instanceof*-Operator auf ein passendes Inventar prüfen sollten, war es nicht möglich das passende Inventar als Attribut zu hinterlegen und die Methode in der Superklasse die Prüfung zu überlassen. Nun haben wir eine Objektattribut `isBeatenBy` vom Typ `Class` eingeführt, mit dessen Hilfe der Vergleich durchgeführt wird. Somit können wir die prüfende Methode in der Superklasse `Obstacle` verschieben. Dadurch haben wir 3-Fach duplizierten Code in nur eine Methode zusammengeführt.

Listing 1: Klasse *Obstacle*

```
3 /**
4  * Oberklasse für alle Hindernisse. Diese Klasse vereint alle Hindernisse,
5  * so dass man leichter auf ein Hindernis prüfen kann.
6  *
7  * @author Beate Ruffer (Bea), Mohamadreza Khostevan (Amir), Leopold Schulz-
8  *         Hanke (Leo)
9  * @version 2.0.0
10 */
11 public class Obstacle extends Actor
12 {
13     /**
14      * Sound Datei, die Abgespielt wird, wenn dieses Hindernisse überwunden
15      * wurde.
16      */
17     public String beatenSound;
18
19     /**
20      * Definiert durch Objekte welcher Klasse das Hindernis überwunden werden
21      * kann
22      */
23     protected Class isBeatenBy;
24
25     /**
26      * Konstruktor von Obstacle wird von der Subklasse aufgerufen.
27      * @param isBeatenBy Die Klasse des Objekts, welches das Hindernis ü
28      *         berwinden kann.
29      */
30     public Obstacle(Class isBeatenBy){
31         this.isBeatenBy = isBeatenBy;
32     }
33 }
```

```

32      * Prüft ob das sich im Inventar befindende Objekt eine Instanz von
33      * isBeatenBy ist. Falls ja, wird das Hindernis von dem Objekt
34      * im Inventar geschlagen und die Heldin kann passieren.
35      *
36      * @param collectable Das zu prüfende Objekt.
37      * @return true, wenn Hindernis überwunden ist.
38      */
39      public boolean isBeaten(Actor collectable)
40      {
41          return collectable.getClass() == isBeatenBy ;
42      }

```

Nun müssen die Klassen der Hindernisse, nur noch das passende Inventar (über den Konstruktor der Superklasse) und den Sound setzen.

Listing 2: Konstruktor von *Stone*

```

18 public Stone(){
19     super(Hammer.class);
20     beatenSound = "explosion.wav";
21 }

```

Tests. Da nun mal jedes kleine Refactoring Regression-Bugs hervorbringen kann, haben wir erneut getestet. Wir haben einen Hammer in unser Inventar aufgenommen, konnten wie erwartet keinen Bush oder Comb passieren. Nur der Stone konnte überwunden werden (der richtige Sound wurde auch abgespielt). Mit der Schere im Inventar konnten wir den Stone nicht überwinden, dafür aber Comb und Bush (ebenfalls mit den richtigen Sounds).

Aufgabe 2 Massenkarambolage (30%)

1. Bla bla bla

Listing 3: *getCollidingObject(Class cls)*-Methode

```

25 /**
26     * Liefert das Objekt der Klasse cls mit dem eine Kollision besteht
27     *
28     * @param cls die Klasse, nach dessen Kollision geprüft wird
29     * @return das Objekt, mit dem kollidiert wurde
30     */
31     public Actor getCollidingObject(Class cls){
32         return getOneIntersectingObject(cls);
33     }

```

2. Bla bla bla

Listing 4: *collidesWith(Class cls)*-Methode

```

25 /**
26     * Prüft ob gerade eine Kollision besteht
27     *
28     * @param cls die Klasse, nach dessen Kollision geprüft wird
29     * @return ob gerade eine Kollision mit einer Instanz der Klasse cls besteht
30     */

```

```
30     public boolean collidesWith(Class cls){
31         return getCollidingObject(cls) != null;
32     }
```

3. Bla bla bla

Listing 5: *newCollisionWith(Class cls)*-Methode

```
25  /**
26   * Prüft, ob gerade eine neue Kollision mit einer Instanz
27   * der Klasse cls begonnen hat.
28   * @param cls die Klasse, nach dessen Kollision geprüft wird
29   * @return ob gerade eine neue Kollision mit einer Instanz der
30   *         Klasse cls begonnen hat
31   */
32  public boolean newCollisionWith(Class cls) {
33      Actor obj = getCollidingObject(cls);
34      if ( collidesWith(cls) ){
35          if (collisions.get(cls) == null || collisions.get(cls) !=
36              obj ){
37              collisions.put(cls, obj);
38              return true;
39          }
40      } else {
41          collisions.remove(cls);
42      }
43      return false;
44  }
```

Aufgabe 3 Richtig kollidieren (50%)

- 1.
- 2.
- 3.