Praktische Informatik I WiSe 2012/2013

Übungsblatt 4

Abgabe: 18.11.2012

Aufgabe 1 Die Welt ist nicht genug (50%)

Unsere HeldIn hat langsam genug davon, immer in derselben Welt herumzulaufen. Sie sucht Abwechslung. Erzeugt deshalb eine neue Welt, die selbst gar keine Spielumgebung darstellt, sondern nur als Weltenverwalter fungiert. Diese Klasse soll ein *Klassenattribut* besitzen, das in einer `ArrayList` die eigentlichen Welten (Level) speichert. Diese Welten werden alle im Konstruktor des Weltenverwalters konstruiert und der `ArrayList` hinzugefügt.[1] Am Ende des Konstruktors wird dann eine dieser Welten als diejenige gesetzt, die zuerst angezeigt werden soll. Diese zuerst angezeigte Welt muss die HeldIn und ihr Inventar enthalten.

Implementiert eine *Klassenmethode*, die ein Argument vom Typ `Class` bekommt und die Welt aus der `ArrayList` zurückliefert, die diesen Typ hat. Falls die gesuchte Klasse nicht vorkommt, soll `null` zurückgeliefert werden. Ihr könnt eure Implementierung testen, indem ihr die Methode im Kontextmenü der Weltenverwalter-Klasse aufruft und im Greenfoot-Stil z.B. `MyWorld.class` als Parameter übergebt. Beachtet den Typ der zurückgelieferten Referenz.

Damit das ganze Sinn macht, braucht ihr natürlich mindestens eine zweite Spielwelt (Level 2?). In dieser brauchen weder die HeldIn noch ihr Inventar enthalten zu sein.

Tipps. `Greenfoot.setWorld` erlaubt es, programmgesteuert ein neues Objekt als die Welt zu setzen, die gerade angezeigt wird. Greenfoot selbst merkt sich eine Welt als Hauptwelt, d.h. die Klasse, deren Instanz nach dem Übersetzen oder dem Drücken auf *Reset* konstruiert wird. Diese Welt ist diejenige, die ihr zuletzt mit Hilfe ihres Kontextmenü-Eintrags „new ...“ konstruiert hat, d.h. durch die Verwendung des Kontextmenüeintrags könnt ihr die Hauptwelt festlegen. Ob ein Objekt eine Instanz einer bestimmten Klasse ist, könnt ihr im Greenfoot-Stil feststellen, indem ihr eine Methode `boolean isOfClass(World w, Class c) {return w.getClass() == c;}` definiert und diese z.B. mit `isOfClass(myWorldObject, MyWorld.class)` benutzt[2].

Aufgabe 2 Wanderer der Welten (50%)

Erzeugt eine Unterklasse von `Actor`, die Ausgänge zu anderen Welten repräsentiert. Ein Ausgang speichert drei Attribute, nämlich die Klasse der Welt, in der die HeldIn landet, wenn sie den Ausgang berührt, und ihre x- und y-Koordinaten, an denen sie in der neuen Welt auftaucht. Diese Werte werden dem Konstruktor des Ausgangs als Parameter übergeben und seine Attribute damit initialisiert.

Wenn der Ausgang eine Kollision mit der HeldIn feststellt, muss er diese mit ihrem Inventar aus der aktuellen Welt entfernen. Dann fragt er den Weltenverwalter nach der Instanz der Klasse der

[1] Es bietet sich an, auch die `ArrayList` selbst erst im Konstruktor zu erzeugen, damit sie bei jeder Erzeugung eines Weltenverwalters durch Greenfoot neu gesetzt wird. Ansonsten könnten sich noch alte Welten in ihr befinden.

[2] Anders als bei Greenfoot üblich, wird so allerdings Vererbung ignoriert.

Zielwelt, fügt dort die HeldIn und ihr Inventar wieder ein (letzteres vielleicht an einer immer gleichen Position) und setzt dann die Zielwelt als die aktuell angezeigte.

Tipp. Vermutlich ist es günstig, wenn der Ausgang die HeldIn nach ihrem Inventar fragen kann, damit er dieses mit in die neue Welt transferieren kann. Obwohl `getOneIntersectingObject(HeldIn.class)` bei einer Überschneidung ja ganz klar eine Instanz der Klasse `HeldIn` zurückgibt, ist der Typ der zurückgelieferten Referenz leider `Actor`. Eine *explizite Typumwandlung* behebt das Problem: `(HeldIn) getOneIntersectingObject(HeldIn.class)`.