Projekt Pushy – Übung zu Greenfoot

Das Projekt Pushy kann im Anschluss an das Projekt Schatzräuber selbständig durchgearbeitet werden. Bei dem Spiel handelt es sich um eine vereinfachte Form des Spiels „Sokoban“.

Wichtige Inhalte sind:

- Vererbung,
- Typecasting,
- der Parameter this.

Aufgabe 1

Machen Sie sich mit dem Spiel anhand der Datei `Pushy.jar` vertraut. Ihre Aufgabe ist es, die Spielfigur Pushy durch ein Labyrinth zu steuern und Diamanten einzusammeln. Im Labyrinth befinden sich Kisten, die Hindernisse für Pushy darstellen. Pushy kann einzelne Kisten verschieben. Wenn Pushy alle Diamanten aufgesammelt hat, muss er noch den Ausgang erreichen.

Aufgabe 2

a) *Kopieren Sie sich das Szenario* `PushyRohling` *und öffnen Sie es mit Greenfoot.*

In der Klasse `PushyWorld` sind schon mehrere Welten angelegt. Im Konstruktor sollte zu diesem Zeitpunkt die Methode `welt0()` aufgerufen werden.

Die Spielfigur ist eine Instanz der Subklasse `Pushy` *von* `Movable`. *Steuern Sie die Spielfigur durch Aufrufe der Methode* `public void move(char richtung)` *der Superklasse* `Movable`.

b) `Pushy` *soll jetzt mit den Cursortasten gesteuert werden. Implementieren Sie die Methode* `private void bewegen()` *in der Klasse* `Pushy`.

Als Vorlage kann Ihnen die entsprechende Methode für den Abenteuer im Projekt Schatzräuber dienen.

Aufgabe 3

a) *Arbeiten Sie weiter mit dem Szenario* `PushyRohling`. *Ändern Sie im Konstruktor von* `PushyWorld` *den Methodenaufruf von* `welt0()` *zu* `welt1()`.

Neben Pushy befinden sich jetzt ein Diamant und der Ausgang in der Welt.

b) *Implementieren Sie die Methode* `private void aufsammeln()` *in der Klasse* `Pushy`.

Mit dieser Methode soll Pushy die Diamanten in der Welt aufsammeln. Als Vorlage kann Ihnen die entsprechende Methode für den Abenteuer im Projekt Schatzräuber dienen.

c) Wenn Pushy das Feld betritt, in dem sich der Ausgang befindet, soll Pushy die Welt verlassen. Dies kann bis auf ganz kleine Änderungen ähnlich zum Aufsammeln von Diamanten implementiert werden. Die Methode zum Entfernen eines Objekts aus der Welt ist die gleiche. Als Parameter muss jetzt aber das Pushy-Objekt selbst übergeben werden. Dieser Parameter wird mit dem Schlüsselwort „`this`“ übergeben.

Implementieren Sie die Methode `private void beenden()` *in der Klasse* `Pushy`.

d) Wenn Pushy die Welt verlassen hat, muss in der Klasse `PushyWorld` entschieden werden, ob Pushy erfolgreich war oder nicht. In Abhängigkeit davon wird das Schlussbild gesetzt. Pushy war erfolgreich, wenn er alle Diamanten eingesammelt hat und die Welt verlassen hat.

Implementieren Sie dies in der `act()-Methode` *der Klasse* `PushyWorld`.. *Vorlage ist wieder das Projekt* `Schatzräuber`.

Aufgabe 4

a) *Arbeiten Sie weiter mit dem Szenario* `PushyRohling`. *Ändern Sie im Konstruktor von* `PushyWorld` *den Methodenaufruf von* `welt1()` *zu* `welt2()`.

Zusätzlich zu Pushy, dem Diamanten und dem Ausgang gibt es jetzt noch eine Reihe von Kisten in der Welt. Pushy muss einzelne Kisten in seine momentane Bewegungsrichtung verschieben können, um seine Aufgabe zu erledigen. Dies wird im Folgenden implementiert.

b) *Informieren Sie Sich in der Greenfoot Klassendokumentation über die Methode* `getOneObjectAtOffset()` *der Klasse* `Actor`.

c) Im Folgenden verändern Sie die Methode bewegen() in der Klasse `Pushy`. In jeder der vier if-Abfragen lassen Sie sich mit der Methode `getOneObjectAtOffset()` ein Kistenobjekt geben. Falls ein solches Objekt nicht existiert, erhalten Sie „`null`“ zurück.

Versuchen Sie es zunächst mit dem Aufruf

```
Kiste k = getOneObjectAtOffset(1, 0, Kiste.class);
```

falls Pushy nach rechts geht.

Beim Übersetzen erhalten Sie die Fehlermeldung: `incompatible types.` Dies liegt daran, dass sie auf der linken Seite der Zuweisung eine `Kiste` deklarieren, die Methode `getOneObjectAtOffset()` aber einen `Actor` als Rückgabewert definiert. Die Klasse `Kiste` ist eine `Subklasse` von `Actor` und damit auch ein `Actor`. Der Compiler kann dies aber nicht wissen. Daher muss man es explizit mitteilen. Setzen Sie zu diesem Zweck den Klassennamen in Klammern vor den Methodenaufruf:

```
Kiste k = (Kiste)getOneObjectAtOffset(1, 0, Kiste.class);
```

Dies wird `Typecasting` genannt. (Anmerkung: Der Begriff `Typecasting` beinhaltet deutlich mehr als hier beschrieben. Fürs Erste genügt aber diese Erklärung.)

Überprüfen Sie anschließend, ob das erhaltene Objekt ungleich `null` ist. Der Operator für ungleich ist „`!=`“. Falls k nicht gleich `null` ist, können sie mit der Punktnotation an dem Objekt k die Methode move() aufrufen:

```
k.move('r');
```

Info: Im Projekt `Pushy` ist das Spiel komplett implementiert. In der Klasse `PushyWorld` gibt es vier Welten mit unterschiedlichem Schwierigkeitsgrad.

Viel Spaß!

Übrigens: Im Original des Spiels gibt es noch farbige Kugeln, die Pushy auf Felder der gleichen Farbe in der Welt verschieben muss. Wenn eine Kugel auf dem richtigen Feld zu liegen kommt, kann Pushy die Kugel nicht mehr bewegen.

Falls Sie also noch Zeit und Lust haben … ;-)