Übungsblatt 3

Musterlösung

Aufgabe 1 Einmal mit Klasse (20 %)

Die Klasse *Walker* enthält die drei geforderten Attribute. Sie sind alle privat, also von außen nicht zugreifbar. Das Spielobjekt und die maximale Anzahl der Schritte bleiben über die Lebenszeit einer Instanz der Klasse unverändert, sind also konstant. Die bisher gelaufene Anzahl von Schritten muss hingegen weitergezählt werden, ist also veränderlich.

Achtung: Ohne den Konstruktor aus Aufgabe 2 kann man den Quelltext nicht übersetzen, da den beiden Konstanten noch kein Wert zugewiesen wurde.

```
10 class Walker
11 {
12     private final GameObject avatar;
13     private final int maxSteps;
14     private int stepsSoFar;
57 }
```

Aufgabe 2 Konstruktivismus (20 %)

Der Konstruktor bekommt die gleichen Parameter wie die Attribute und initialisiert diese damit. Da der Zweck der Parameter des Konstruktors nur die Initialisierung der Attribute ist, sind sie alle konstant.[1]

```
16     Walker(final GameObject avatar, final int maxSteps, final int stepsSoFar)
17     {
18         this.avatar = avatar;
19         this.maxSteps = maxSteps;
20         this.stepsSoFar = stepsSoFar;
21     }
```

Aufgabe 3 Schritt für Schritt (40 %)

Die Methode *act()* bewegt das Spielobjekt zunächst einen Schritt in Richtung seiner Orientierung. Dazu wird zwischen den vier möglichen Rotationen unterschieden und für jeden Fall die Position geeignet geändert. Danach wird ein Schritt-Sound abgespielt, da sich die Figur ja auf jeden Fall bewegt hat.

```
23     void act()
24     {
25         // Vorwärts bewegen
26         if (avatar.getRotation() == 0) {
27             avatar.setLocation(avatar.getX() + 1, avatar.getY());
28         }
29         else if (avatar.getRotation() == 1) {
30             avatar.setLocation(avatar.getX(), avatar.getY() + 1);
```

[1] Eigentlich wäre es schön, wenn an dieser Stelle geprüft würde, ob *maxSteps* größer als 0 ist und *stepsSoFar* mindestens 0 und kleiner als *maxSteps* ist, denn anderenfalls wird sich ein Objekt später nicht wie gedacht verhalten. Fehlerbehandlung wird noch ein Thema in der Veranstaltung sein.

```
31          }
32          else if (avatar.getRotation() == 2) {
33              avatar.setLocation(avatar.getX() - 1, avatar.getY());
34          }
35          else {
36              avatar.setLocation(avatar.getX(), avatar.getY() - 1);
37          }
38
39          // Sound dazu abspielen
40          avatar.playSound("step");
```

Dann wird der Schrittzähler um 1 erhöht. Erreicht er dabei die maximale Anzahl der Schritte, wird die Spielfigur um 180° gedreht und der Schrittzähler wieder auf 0 zurückgesetzt.

```
42          // Weiterzählen
43          stepsSoFar = stepsSoFar + 1;
44
45          // Wenn maximale Anzahl erreicht, umdrehen und Zählung neu beginnen
46          if (stepsSoFar == maxSteps) {
47              avatar.setRotation(avatar.getRotation() + 2);
48              stepsSoFar = 0;
49          }
```

Aufgabe 4 Und Action! (20 %)

In der Klasse *PI1Game* wurden die folgenden Zeilen geändert, so dass die *GameObject*-Instanzen gleich an neue *Walker*-Objekte übergeben werden.

```
37          final Walker walker1 = new Walker(new GameObject(1, 0, 2, "claudius"), 3, 2);
38          final Walker walker2 = new Walker(new GameObject(0, 1, 0, "laila"), 3, 0);
39          final Walker walker3 = new Walker(new GameObject(3, 2, 2, "child"), 4, 1);
```

Da es erforderlich ist, dass sich die NPC nur bewegen, wenn sich die Spielfigur auch bewegt hat, wird die Ausführung des Schleifenrumpfs vorzeitig abgebrochen, wenn eine ungültige Taste gedrückt wurde.

```
60              else {
61                  playSound("error");
62                  continue;
63              }
```

Dadurch kann dann im Anschluss alles ausgeführt werden, was nur nach einem gültigen Schritt folgen sollte. Das Abspielen des Schrittsounds wurde auch hierhin verschoben. Es wird noch kurz gewartet, damit sich die anderen Figuren tatsächlich etwas später als die Spielfigur bewegen.

```
65              playSound("step");
66              sleep(200);
67              walker1.act();
68              wlaker2.act();
69              walker3.act();
```

Aufgabe 5 Bonusaufgabe (5 %)

Es gibt mehrere Möglichkeiten, wie die NPC-Klasse an die Position der Spielfigur kommen kann, z.B. könnte die Spielfigur zusätzlich dem Konstruktor übergeben werden, der sie für den späteren Zugriff in einem Attribut speichert. Hier wurde aber die einfachere Variante gewählt, die Spielfigur direkt an die Methode *act* als Parameter zu übergeben.

```
23      void act(final GameObject player)
```

Am Ende der Methode wird dann geprüft, ob sowohl die x- als auch die y-Koordinate gleich sind. Wenn ja, verschwindet die Spielfigur und es wird ein Sound abgespielt.[2]

```
52          if (avatar.getX() == player.getX() && avatar.getY() == player.getY()) {
53              player.setVisible(false);
54              avatar.playSound("go-away");
55          }
```

In *PI1Game* hängt nun die Fortsetzung der Schleife von der Sichtbarkeit der Spielfigur ab.

```
42          while (player.isVisible()) {
```

Außerdem muss die Spielfigur in der Klasse *PI1Game* dann natürlich noch als aktueller Parameter an die *act*-Methoden übergeben werden.

```
67              walker1.act(player);
68              walker2.act(player);
69              walker3.act(player);
```

[2] Die Lizenzinformation wurde der Datei *README.TXT* hinzugefügt.