Praktische Informatik I
Tutor:in: Leander Staack

WiSe 2021/22
Bearbeiter:in: Julius Walczynski [6113829]

Übungsblatt 3

Lösungsvorschlag

Aufgabe 1 Einmal mit Klasse

Die Klasse "NPC"deklariert zum Anfang die drei in der Aufgabe verlangten Attribute.

```
6 class NPC {
7     private final GameObject gameObject;
8     private final int walkDistance;
9     private int progress;
```

gameObject	bietet die Referenz zum visuell sichtbaren *GameObject* im *JPanel*.
walkDistance	gibt die Länge des vom *NPC* zu laufenden Weg an.
progress	beschreibt wie viele Schritte der *NPC* vom letzten Endpunkt seiner Route zurückgelegt hat.

Wenn man davon ausgeht, dass *walkDistance* und das *GameObject* des *NPC*s im laufenden Prozess nicht verändert werden, können diese als *final* bzw. konstant markiert werden.

Aufgabe 2 Konstruktivismus

```
11     public NPC(final int _x, final int _y, final String _fileName, int _walkDistance, int
           _progress){
12         gameObject = new GameObject(_x, _y, 0, _fileName);
13         walkDistance = _walkDistance;
14         progress = (_progress>walkDistance||_progress<1)?1:_progress;
15     }
```

Der Konstruktor initialisiert die drei Attribute aus Aufgabe 1.
Hierzu nimmt er 5 Parameter entgegen:

_x, *_y*, *_fileName*	werden an den Konstruktor des *gameObject*s weitergegeben, um dieses zu initialisieren.
_walkDistance, *_progress*	definieren die dazugehörigen Attribute des *NPC*-Objekts.

Bevor der Wert des Parameters *_progress* in *progress* geschrieben wird, wird sichergestellt, dass der Parameter im sinnvollen Rahmen der Anwendung liegt. D.h. sollte *_progress*< 1 oder *_progress*$>$*walkDistance* sein, wird er vorsichtshalber auf 1 zurückgesetzt. (Hierbei besitzen die Start/Endfelder der Route einen *progress*-Wert von 1 bzw. *walkDistance*)

Aufgabe 3 Schritt für Schritt

```
17      public void act(){
18          progress++;
19          if(progress>=walkDistance){
20              returning = !returning;
21              progress = 1;
22          }
23          int _direction = returning?2:0;
24          Move(_direction);
25      }
```

Beim Ausführen der Funktion *act* wird zuerst *progress* um 1 erhöht. Sollte $progress \geq walkDistance$ sein, ist der *NPC* an einem Start/Endpunkt seiner Route angekommen. Folgend wird *progress* auf 1 zurückgesetzt und die boolean Variable *returning* negiert (geflipt). Anschließend wird anhand von *returning* die Richtung des *NPC*s bestimmt und dieser durch die Funktion *Move* um 1 Feld in die angegebene Richtung bewegt und rotiert.

```
27      private void Move(int _direction){
28          gameObject.setRotation(_direction);
29          int _x = gameObject.getX();
30          int _y = gameObject.getY();
31          switch(_direction){
32              case 0:
33                  _x++;
34                  break;
35              case 1:
36                  _y++;
37                  break;
38              case 2:
39                  _x--;
40                  break;
41              case 3:
42                  _y--;
43                  break;
44          }
45          gameObject.setLocation(MinMax(0, RPGGame.FIELD_X-1, _x), MinMax(0, RPGGame.
              FIELD_Y-1, _y));
46      }
```

Der Parameter *_direction* beihaltet die Richtung, in die sich der *NPC* bewegen soll. Die Richtungskodierung ist der *GameObject*-Klasse in Bezug auf die Rotation des *GameObjects* entnommen. Über ein switch-Statement wird die kodierte Richtung zu einer Bewegung auf der X-Achse und Y-Achse umgewandelt. Bevor die Location des *GameObject*s überschrieben wird, werden die Werte durch die Funktion *MinMax* auf die Dimensionen des Spielbretts begrenzt. D.h $x < 0 \rightarrow 0, x > \text{FIELD\_X} - 1 \rightarrow \text{FIELD\_X} - 1, y < 0 \rightarrow 0, y > \text{FIELD\_Y} - 1 \rightarrow \text{FIELD\_Y} - 1$.

Aufgabe 4 Und Action!

```
74          NPCs[0] = new NPC(1,0, "claudius", 3, 1);
75          NPCs[1] = new NPC(4,1, "child", 5, 4);
```

In der *RPGGame*-Klasse werden zu Beginn des Spiels 2 neue *NPC*s erstellt, deren *act* Methode nach jedem Zug des Spielers aufgerufen werden:

```
114             for(NPC _npc: NPCs){
115                 _npc.act();
```

Aufgabe 5 Bonusaufgabe

```
114             for(NPC _npc: NPCs){
115                 _npc.act();
116                 if(checkCollision(player.getX(), player.getY(), _npc.getX(), _npc.getY())
                      ){
117                     player.setVisible(false);
118                     break gameloop;
119                 }
120             }
```

Nach jedem Zug des Spielers, werden die *NPC*s bewegt. Sollte die neue Position einer der *NPC*s mit der des Spielers übereinstimmen, wird die Spielerfigur unsichtbar gemacht und der GameLoop verlassen.