Praktische Informatik I | WiSe 2023/24
Tutor:in: Pascal Himmelmann | Bearbeiter:in: Öykü Koç

Übungsblatt 10

Abgabe: 14.01.2024

Aufgabe 1 Lässig Level laden

Ich habe eine Level-Klasse erstellt, die es erlaubt, die Beschreibung des Bereichs auf der Karte aus der Datei zu laden, nicht direkt aus einem String im Code. Ich habe die Namen und Richtungssymbole der Player und Walker im Konstruktor definiert und sie der LinkedHashMap hinzugefügt.

Da es insgesamt 4 Richtungen gibt, musste ich für jedes Zeichen unterschiedliche Symbole auf der Karte verwenden:
0 = rechts,
1= unten,
2= links
und 3= oben.
Bei der Auswahl dieser Symbole ging ich von den ASCII-Codes der Buchstaben aus, die ich angegeben hatte;
0,1,2,3 für Frau (Spieler)
@,A,B,C für Claudius (Wanderer)
D,E,F,G für Laila (Spaziergängerin)
H,I,J,K für Kind (Wanderer)
Ich habe die Symbole ausgewählt.

Das Ergebnis der Operation dieser Symbole (ASCII-Code & 3) liefert die folgenden Richtungsergebnisse. Zum Beispiel: Der ASCII-Code des @-Symbols ist 64. Daher ergibt die Operation (64 & 3) die Richtungsangabe "0". Gleichzeitig entsprechen die Richtungszahlen (0,1,2,3) den Indizes der Daten (@ABC) in der VALUE-Information der LinkedHashMap, die die Symbole enthält. Der Vergleich zwischen den Symbolen in der Map und den Symbolen in der LinkedHashMap wird also direkt über den Index vorgenommen.
@=0 ,A=1, B=2 , C=3
D=0, E=1, F=2, G=3
H=0, I=1, J=2, K=3
Mit diesen Ergebnissen konnte ich leicht auf die jeweiligen Richtungen der Zeichen zugreifen.

```
10 /**
11  * Diese Klasse ermöglicht die Erstellung der Klasse Field und die Erstellung der Klassen
            Player und Walker.
12  * Die Karte wird mit der Klasse Field erstellt. Die Spieler werden mit den Klassen
          Player und Walker erstellt.
13  * Sie nimmt den Dateinamen, unter dem die Karte gespeichert ist, als Parameter entgegen.
14  *
15  * @author Öykü Koç
16  */
17 class Level
18 {
19     /** Standardbereichssymbol der Karte. */
20     final private char defaultMapChar = '0';
21
22     /** Array map, mit dem wir die Karte erstellen können */
23     final private String[] map;
```

```
24
25      /** Feldklasse, mit der wir die Karte erstellen können */
26      final private Field field;
27
28      /** LinkedHashMap, in der wir die Symbole und Namen der Akteure speichern. */
29      final private LinkedHashMap<String, String> symbols;
30
31      /** Array List, in der wir die Akteure aufbewahren */
32      final public ArrayList<Actor> actors = new ArrayList<Actor>();
33
34      /** Die Spielfigur. */
35      private Player player;
36
37      /**
38       * Konstrukteur der Klasse Level.
39       * @param fileName Es handelt sich um die Variable mit dem Dateinamen,
40       * die die Symbole der Karte, des Hauptakteurs und der Akteure enthält,
41       * die zur Erstellung einer Karte in der Level-Klasse erforderlich sind.
42       */
43      public Level(String fileName){
44          //Definiere Symbole, die die Namen und Richtungen von Spielern und NPCs enthalten
                .
45          this.symbols = new LinkedHashMap<String, String>();
46          this.symbols.put("woman", "0123");
47          this.symbols.put("claudius", "@ABC");
48          this.symbols.put("laila", "DEFG");
49          this.symbols.put("child", "HIJK");
50
51          // Die aus der Datei gelesenen Kartendaten werden der Map Array List hinzugefügt.
52          this.map = getMapFromFile(fileName);
53
54          // Map Array List wird an die Klasse Field gesendet, um eine Map zu erstellen.
55          this.field = new Field(map);
56
57          // Spieler und Walker-Akteure werden erstellt.
58          createActors();
59      }
```

Dann habe ich die von mir erstellte Datei "Level.txt"gelesen. In dieser Datei habe ich eine Methode geschrieben, um die Zeichen nacheinander zu lesen. Diese Methode hat die Zeilen durchlaufen und mir die Informationen der Karte als String zurückgegeben. Als Kontrollmechanismus habe ich das "#Zeichen am Anfang jeder Zeile verwendet, um es in der split("#")-Methode von String-Daten zu verwenden. Dann habe ich mit Hilfe der split-Methode die String-Daten in ein String[]-Array umgewandelt und das Array zurückgegeben. Ich habe auch Codes geschrieben, die eine Fehlermeldung ausgeben, wenn die als Parameter übergebene Datei "Level.txt"nicht gefunden oder gelesen werden kann.

```
61      /**
62      * In der Klasse Level die Methode, die die Kartendaten als String[] aus dem "in der
                Anwendung angegebenen Dateinamen" an den Benutzer zurückgibt,
63      * indem sie die Zeilen der gewünschten Karte in der Datei nacheinander abfährt.
64      * @param fileName Die Map, die benötigt wird, um eine Map in der Level-Klasse zu
                erstellen,
65      * die die Symbole des Hauptakteurs und der Akteure enthält ist die Variable des
                Dateinamens
66      */
67      private String[] getMapFromFile(String fileName)
68      {
69          int charNum = -1;
70          char character;
71          String fileText = "";
72          InputStream file = null;
73
74          // Fehlermeldung, wenn die Level-Datei nicht gefunden wurde.
75          try{
76              file = Game.Jar.getInputStream(fileName);
77
78          }catch(FileNotFoundException ex){
79
80              if(file == null)
```

```
81                  throw new IllegalArgumentException("Die Level-Datei wurde nicht gefunden.
                            ");
82          }
83
84          //Fehlermeldung beim Einlesen der Datei.
85          try{
86               charNum = file.read();
87
88               if(charNum == -1)
89                    throw new IllegalArgumentException("Es gab Probleme beim Lesen der Datei.
                            ");
90          }
91          catch (IOException  oe )
92          {
93               System.out.println( oe );
94          }
95
96          try
97          {
98
99               while(!(charNum == -1))
100              {
101                   character = (char)charNum;
102
103                   // 10 = Neue Zeile, für die Verwendung in der Methode split() wird das
                            Symbol "#" verwendet, um den Zeilenanfang zu kennzeichnen.
104                   if(charNum == 10)
105                        fileText  = fileText + '#';
106                   else
107                        fileText  = fileText + character;
108
109                   charNum = file.read();
110
111              }
112
113              file.close();
114
115         }catch(IOException  oe ){
116              System.out.println( oe );
117         }
118         finally{
119              // Die gelesene Datei wird dem String[] map Array mit der Methode split()
                        hinzugefügt.
120              String[] map = fileText.split("#");
121              return map;
122         }
123
124    }
```

Eine weitere Methode von mir ist die Methode, die prüft, ob die Symbole in der Map-Datei mit den Symbolen übereinstimmen, die ich für die Player und Walker definiert habe. Diese Methode gibt im Falle einer Übereinstimmung die Namen der Spieler- und Walker-Akteure zurück. Wenn es keine Übereinstimmung gibt, gibt sie den Wert Null zurück.

```
126     /**
127      * Methode zum Abrufen von Akteursnamen, die in der LinkedHashMap named symbols
                gespeichert sind.
128      * Der ASCII-Wert der Symbole aus der Karte definiert sowohl die Richtung als auch
                den Index des Symbols in der LinkedHashMap VALUE.
129      * Wenn die char-Daten in den Symbolen mit den char-Daten aus der Karte ü
                bereinstimmen, wird der Akteurname aus dem KEY zurückgegeben.
130      * @param charNum Der Wert der Zeichennummern der NSC- und Akteurssymbole aus der Map
                , die wir anhand ihrer ASCII-Werte ausgewählt haben.
131      */
132     private String getActorName(int charNum){
133        String result = null;
134
135        for(Map.Entry<String, String> actor : this.symbols.entrySet()){
136
137             if(actor.getValue().charAt(charNum & 3) == (char)charNum){
138                  result = actor.getKey();
139                  break;
```

```
140             }
141
142         }
143
144         return result;
145     }
```

Ich habe eine Methode namens createPlayer erstellt, um das Player-Objekt zu erstellen. Diese Methode prüft dank der Symbole und ASCII-Zahlen, die ich im Constructor definiert habe, ob ein Player im String[] map Array vorhanden ist, und wenn dies der Fall ist, erstellt sie einen neuen Player, indem sie die x- und y-Position übernimmt und ihn der Actor Array List hinzufügt. Wenn unter den gegebenen Bedingungen kein Player vorhanden ist, wird eine Fehlermeldung ausgegeben

```
147     /**
148      * Die Symbole aus der Karte werden mit den in LinkedHasMap definierten Symbolen
                verglichen
149      * Wenn die für den Player definierten Symbole mit dem Symbol aus der Karte ü
                bereinstimmen, wird der Player erstellt.
150      * Der erstellte Player wird der Actor Array List hinzugefügt.
151      */
152     private void createPlayer(){
153         int charNum = 79;
154         String actorName = null;
155
156         for(int y=0; y < map.length; y=y+2){
157
158             for(int x=0; x < map[y].length(); x=x+2){
159                 charNum = (int)map[y].charAt(x);
160
161                 actorName = getActorName(charNum);
162
163                 if(actorName == "woman"){
164                     player = new Player(x/2, y/2, charNum & 3, field);
165                     actors.add(0, player);
166                     break;
167                 }
168
169             }
170          }
171
172         // Fehlercode, wenn der Spieler nicht auf der Karte gefunden werden kann.
173         if(player == null)
174             throw new IllegalArgumentException("Die Datei enthält nicht genau einen
                    Player.");
175
176     }
```

Ich habe eine Methode namens createWalker erstellt, um das Walker-Objekt zu erstellen. Diese Methode prüft, ob ein Walker im String[] map Array vorhanden ist, und wenn dies der Fall ist, erstellt sie ein neues Walker-Objekt, indem sie die x- und y-Position und den Walker-Namen mit der getActorName-Methode abruft und ihn dank der Symbole und ASCII-Zahlen, die ich im Constructor definiert habe, der Actor Array List hinzufügt. Wenn die Symbole im String[] map Array nicht mit den in LinkedHashMap definierten Symbolen übereinstimmen, gebe ich eine Fehlermeldung aus.

```
178     /**
179      * Die Symbole aus der Karte werden mit den in LinkedHasMap definierten Symbolen
                verglichen.
180      * Wenn die für den Walker definierten Symbole mit dem Symbol aus der Karte ü
                bereinstimmen, wird der Walker erstellt.
181      * Der erstellte Walker wird der Actor Array List hinzugefügt.
182      */
183     private void createWalker(){
184         int charNum;
185         String actorName = null;
186
187         for(int y=0; y < map.length; y=y+2){
188             for(int x=0; x < map[y].length(); x=x+2){
```

```
189
190                 charNum = (int)map[y].charAt(x);
191
192                 actorName = getActorName(charNum);
193
194                 if(actorName != null && actorName != "woman")
195                     actors.add(new Walker(x/2, y/2, charNum & 3, actorName, field, player
                                ));
196
197                 // Fehlercode für ungültige Symbole, wenn der Akteursname leer ist und
                        sich vom Standardzuordnungszeichen ('0') unterscheidet.
198                 if(actorName == null && (char)charNum != defaultMapChar)
199                     throw new IllegalArgumentException("Die Datei enthält ungültige
                                Symbole an den Positionen von Gitterknoten.");
200
201             }
202         }
203     }
```

Gegen Ende habe ich eine allgemeine Methode namens createActors geschrieben, die die createPlayer- und createWalker-Methoden ausführt und meine Actor Array List namens getActors erstellt.

```
205     /**
206      * Methode zur Erstellung von Spielern und Walkern.
207      */
208     private void createActors(){
209         createPlayer();
210         createWalker();
211     }
212
213     /**
214      * Methode, die die erzeugte Akteursliste zurückgibt.
215      */
216     public ArrayList<Actor> getActors(){
217         return actors;
218     }
```

In der Methode Main PI1Game habe ich die Klasse Level und zusätzliche Objekte erstellt und die Methode act() aufgerufen, um die Akteure in der While-Schleife zu aktivieren

```
4  /**
5   * Dies ist die Hauptklasse eines Spiels. Sie enthält die Hauptmethode, die zum
6   * Starten des Spiels aufgerufen werden muss.
7   *
8   * @author Öykü Koç
9   */
10 abstract class PI1Game extends Game
11 {
12     /** Das Spiel beginnt durch Aufruf dieser Methode. */
13     static void main()
14     {
15         // Erzeugt die Klasse Level.
16         Level level = new Level("Level.txt");
17
18         // Gibt den Spieler in der Liste Akteur zurück.
19         Player player = (Player)level.getActors().get(0);
20
21         // Array List, in der die der Karte hinzugefügten Objekte gespeichert werden.
22         ArrayList<GameObject> gameObjects = new ArrayList<GameObject>();
23         gameObjects.add(new GameObject(3, 0, 0, "goal"));
24         gameObjects.add(new GameObject(4, 0, 0, "bridge"));
25         gameObjects.add(new GameObject(4, 1, 3, "water-l"));
26         gameObjects.add(new GameObject(3, 3, 0, "water-l"));
27         gameObjects.add(new GameObject(4, 3, 0, "water-i"));
28
29         while (player.isVisible()) {
30             for (final Actor actor : level.getActors()) {
31                 // Ruft die Bewegungsmethode aller Actors auf.
32                 actor.act();
33
```

```
34                  // Schleife, die sicherstellt, dass bei der Zerstörung des Players auch
                            alle anderen Akteure zerstört werden
35                  if(player.isVisible()==false){
36                      level.hide();
37
38                      for(GameObject obj : gameObjects){
39                          obj.setVisible(false);
40                      }
41                  }
42
43              }
44          }
45      }
46  }
```

Aufgabe 2 Bonusaufgabe: Ich bin dann mal weg

In der Klasse Level habe ich eine hide()-Methode geschrieben, um die erstellten Akteure zu zerstören. In der Klasse Field habe ich eine ArrayList geschrieben, die die erstellten Bodenobjekte der Karte enthält, und eine hide()-Methode, die die Objekte in dieser Liste zerstört. Und ich habe diese Methode in der hide-Methode der Klasse Level verwendet. Schließlich habe ich in der PI1 Main Klasse, wenn Player visible false ist, die öffentliche hide() Methode in der Level Klasse aufgerufen, um den Boden und die Akteure zu zerstören. Ich habe auch alle Objekte zerstört, indem ich die Objekte in der gameObjects Array List, die die zusätzlichen Objekte enthält, die in PI1 Main erstellt wurden, in der for Schleife zerstört habe.

```
220     /**
221      * Methode zum Zerstören aller Spieler-, Walker- und Kartenobjekte.
222      */
223     public void hide(){
224        for(GameObject actors : actors){
225          actors.setVisible(false);
226        }
227
228        field.hide();
229     }
```

```
46      /**
47       * Array Liste, in der die Bodenobjekte, aus denen die Karte besteht, gespeichert
                werden.
48       */
49      private final ArrayList<GameObject> gameObjectList = new ArrayList<GameObject>();

57      Field(final String[] field)
58      {
59          this.field = field;
60
61          for (int y = 0; y < field.length; y += 2) {
62              for (int x = 0; x < field[y].length(); x += 2) {
63                  gameObjectList.add(new GameObject(x / 2, y / 2, 0,
                            NEIGHBORHOOD_TO_FILENAME[getNeighborhood(x, y)]));
64              }
65          }
66      }
```

```
134     /**
135      * Methode zur Zerstörung von Bodenobjekten auf der Karte.
136      */
137     public void hide()
138     {
139        for(GameObject obj : gameObjectList){
140          obj.setVisible(false);
141        }
142     }
```

```
21          // Array List, in der die der Karte hinzugefügten Objekte gespeichert werden.
22          ArrayList<GameObject> gameObjects = new ArrayList<GameObject>();
23          gameObjects.add(new GameObject(3, 0, 0, "goal"));
24          gameObjects.add(new GameObject(4, 0, 0, "bridge"));
25          gameObjects.add(new GameObject(4, 1, 3, "water-l"));
26          gameObjects.add(new GameObject(3, 3, 0, "water-l"));
27          gameObjects.add(new GameObject(4, 3, 0, "water-i"));
28
29          while (player.isVisible()) {
30              for (final Actor actor : level.getActors()) {
31                  // Ruft die Bewegungsmethode aller Actors auf.
32                  actor.act();
33
34                  // Schleife, die sicherstellt, dass bei der Zerstörung des Players auch
                        alle anderen Akteure zerstört werden
35                  if(player.isVisible()==false){
36                      level.hide();
37
38                      for(GameObject obj : gameObjects){
39                          obj.setVisible(false);
40                      }
41                  }
42
43              }
44          }
```