Praktische Informatik I
Tutor:in: Leander Staack

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

Übungsblatt 5

Lösungsvorschlag

Aufgabe 1 Zugriffssicherheit

```
38      public Field(String[] _map){
39          this.map=_map;
```

Der Konstruktor der Klasse *Field* nimmt die Beschreibung der Karte als *String[ ] _map* entgegen und speichert sie lokal im Attribut *map* ab.

```
59      public char getCell(int x, int y){
60          if(y<0||x<0||y>=this.map.length||x>=this.map[y].length()){
61              return ' ';
62          }
63          return this.map[y].charAt(x);
64      }
```

Die Methode *getCell(int x, int y)* nimmt eine x- und y-Koordinate entgegen und gibt den dazugehörigen *char* aus *map* zurück. Dabei werden zuerst die x- und y-Werte überprüft. Sollten diese außerhalb des Definitionsbereichs von *map* liegen ($x < 0 \vee y < 0 \vee y \geq$ Anzahl der Elemente von *map* $\vee x \geq$ Anzahl der chars im String *map[y]*), wird ein Leerzeichen (' ') als Defaultwert zurückgegeben.

Aufgabe 2 Nachbarschaftshilfe

```
48      public int getNeighboorhood(int x, int y){
49          return isCellWhitespace(x+1,y)<<0
50                 |isCellWhitespace(x,y+1)<<1
51                 |isCellWhitespace(x-1,y)<<2
52                 |isCellWhitespace(x,y-1)<<3;
53      }
```

Die Methode *getNeighborhood(int x, int y)* nimmt eine x- und y-Koordinate entgegen und gibt die ausgerechnete Nachbarschafts-Signatur zurück. Die Signatur wird berechnet, indem die niedrigsten 4 Bits einer *int* in der Reihenfolge unten, links, oben, rechts auf 1 gesetzt werden, falls die Nachbarzelle in dieser Richtung **kein** Leerzeichen (' ') und auf 0, falls die Nachbarzelle in dieser Richtung **ein** Leerzeichen (' ') ist. Die resultierende 4 Bit Signatur (0b0000 - 0b1111) entspricht dem Index (0-15) des Dateinamens der korrespondierenden Grafik in *NEIGHBORHOOD_TO_FILENAME*.

Aufgabe 3 Feldkonstruktion

```
38      public Field(String[] _map){
39          this.map=_map;
40          for(int y = 0; y < this.map.length; y+=2){
41              for(int x = 0; x < this.map[y].length(); x+=2){
42                  int signature = getNeighboorhood(x,y);
43                  new GameObject(x/2, y/2, 0, NEIGHBORHOOD_TO_FILENAME[signature]);
44              }
45          }
46      }
```

Der Konstruktor konstruiert aus der Spielfeldbeschreibung (*map*) ein Gitter aus *GameObject*s mit den passenden Grafiken. Hierfür tasten 2 *for*-Schleifen die *String*s in *map* und deren *char*s ab. Da die Mitte der Felder nur auf jeder 2. x- bzw. y-Koordinate liegt, werden x und y in jedem Durchlauf der dazugehörigen Schleife um 2 erhöht. Für jedes Feld an den Koordinaten (x,y) wird mit Hilfe der *getNeighborhood* Methode die passende Nachbarschafts-Signatur berechnet, darüber der Dateiname der Grafik aus *NEIGHBORHOOD_TO_FILENAME* entnommen und zusammen mit x,y als Parameter in den Konstruktor neuer *GameObject*s übergeben. Da die beiden Schleifen x und y pro Durchlauf um 2 erhöhen, müssen x und y durch 2 geteilt werden, um sinnvolle Koordinaten für die *GameObject*s darzustellen.