Ein Klassiker: Conways Spiel des Lebens

Sven Eric Panitz

7. Januar 2023

Inhaltsverzeichnis

1 Ziel der Aufgabe

Ziel dieser Aufgabe ist es eine einfache Version von ›Conways Spiel des Lebens‹[Gam80] in Java zu implementieren.

Das Spielfeld wird dargestellt durch Matrizen, deren Einträge entweder belegt (bevölkert), oder nicht belegt sind. Ein Eintrag an Position (i, j) in einer Matrix wird durch seine Zeile i und sein Spalte j bezeichnet, wobei die Zählung der Zeilen und Spalten bei 0 beginnt.

Es gibt nun Regeln, wie aus einer Belegung des Spielfelds die Belegung im nächsten Spielzyklus ist. Es wird nach diesen Regeln die Population der nächsten Generation gebildet.

Damit stellt das Spiel eine kleine Simulation dar.

2 Aufgaben

GameOfLife.java

```
record GameOfLife(boolean[][] field){

```

a) Schreiben Sie eine Funktion die für eine Matrix ein Stringdarstellung erstellt. Das Spielfeld soll in Zeilen eines Strings dargestellt werden. Für belegte Positionen empfielt sich das Zeichen `'\u2588'`, das ein komplett schwarz gefülltes Zeichen ist. Für unbelegte Felder kann man das Leerzeichen verwenden.

GameOfLife.java

```
String show(){
  return "";
}
```

b) Schreiben Sie eine Methode, die angibt, wie groß die Gesamtpopulation ist. Dies ist die Summe aller Felder, in denen der Wert `true` steht.

GameOfLife.java

```
int population(){
  //To Do
  return 0;
}
```

c) Schreiben Sie eine Methode, die testet, ob kein Feld mehr belegt ist, die Population also ausgestorben ist.

GameOfLife.java

```
boolean extinct(){
  //To Do
  return false;
}
```

d) Jetzt soll ein Spielobjekt aus einem String eingelesen werden.

GameOfLife.java

```
static GameOfLife fromString(String m){
  //To Do
}
```

So soll ein String der folgenden Form analysiert werden. Das Zeichen ›.‹ steht für eine Zelle, die nicht bevölkert ist, alle anderen Zeichen für eine bevölkerte Zelle.

GameOfLife.java

```
  static String ex1 =  """
..................
..................
..................
..................
......0....0......
....00.0000.00....
......0....0......
..................
..................
..................
..................""";
```

Zusammen mit der Methode `show` können wir das Ergebnis zum Beispiel mit folgendem Aufruf testen.

```
System.out.println(GameOfLife.fromString(ex1).show())
```

```
  #  #
## #### ##
  #  #
```

Hinweis: Für einen String `t` können Sie mit `t.split("\n")` eine Reihung der Zeilen bekommen. Mit `t.charAt(i)` das Zeichen am Index i.

e) Schreiben Sie eine Funktion die im Spielfeld für ein Feld mit Index x und y berechnet, wieviel der maximal acht benachbarten Felder mit true belegt sind.

GameOfLife.java

```
int anzahlBelegterNachbarn(int x, int y){
  //ToDo
  return 0;
}
```

f) Schreiben Sie eine Funktion, die den nächsten Spielzustand als neues Spielobjekt erzeugt.

Die Ziel-Matrix soll so gefüllt werden, dass sie nach den Regeln des Spiels wie folgt belegt ist:

Für eine Matrix-Zelle, die in der Quellmatrix belegt ist, gilt:

- Jede Zelle mit einem oder keinem Nachbarn stirbt.
- Jede Zelle mit vier oder mehr Nachbarn stirbt.
- Jede Zelle mit zwei oder drei Nachbarn überlebt.

Für eine Matrix-Zelle, die in der Quellmatrix leer ist, gilt:

- Jede Zelle mit genau drei belegten Nachbarn wird belegt.

Hier bedeutet ›stirbt‹ nicht belegt in der Matrix (false) und ›überlebt‹: belegt in der Matrix (true).

GameOfLife.java

```
GameOfLife nextGeneration(){
}
```

g) Mit einem kleinen Trick, können wir das Spiel in der JShell durchspielen.

Mit dem String `"\033[H\033[2J"` lässt sich der Bildschirm komplett frei machen und der Positionszeiger ganz nach oben setzen.

So können wir eine Funktion definieren, die kurz verzögert und dann die Bildschirm wieder frei macht.

GameOfLife.java

```
void waitAndClear(){
  try{Thread.sleep(2000);}catch(Exception e){}
  System.out.print("\033[H\033[2J");
  System.out.flush();
}
```

Schreiben Sie eine Funktion `play`, die das Spiel endlos durchiteriert.

GameOfLife.java

```
void play(){
  var current = this;
}
```

Schreiben Sie hierzu eine Schleife, die so lange immer wieder, `current` anzeigt, dann etwas wartet und den Bildschirm löscht und dann `current` auf die nächste generation setzt, bis `current` ausgestorben ist.

Sie sollten jetzt in der Lage sein, zum Beispiel mit folgendem Aufruf in der JShell, das Spiel zu starten.

```
jshell> GameOfLife.fromString(GameOfLife.ex1).play()
```

Oder Sie kompilieren die Klasse und verwenden folgende Main-Methode:

GameOfLife.java

```
  public static void main(String... args){
    GameOfLife.fromString(GameOfLife.ex1).play();
  }
}
```

Literatur

[Gam70] Mathematical Games. The fantastic combinations of John Conway's new solitaire game "life" by Martin Gardner. Scientific American, 223:120–123, 1970.