Analyse «Game of Life»

Verteilte Systeme und Komponenten

Gruppe 5 (Bucher, Kiser, Meyer, Sägesser)

26.02.2018

Inhaltsverzeichnis

Allgemein

Bei der vorliegenden Applikation handelt es sich um eine Implementierung von John Horton Comways Simulation *Game of Life* als Java Applet.

Bei der Simulation gibt es ein Raster mit Zellen, die entweder den Satus aktiv oder inaktiv bzw. lebendig oder tot haben. Zu Beginn der Simulation werden einige lebendige Zellen vordefiniert. Mit jedem Zyklus entstehen, bleiben oder verschwinden die Zellen gemäss folgenden Regeln:

1. Eine lebendige Zelle mit weniger als zwei lebendigen Nachbarzellen stirbt (Unterbevölkerung).
2. Eine lebendige Zelle mit zwei oder drei lebendigen Nachbarzellen überlebt.
3. Eine lebendige Zelle mit mehr als drei lebendigen Nachbarzellen stirbt (Überbevölkerung).
4. Eine tote Zelle mit genau drei lebendigen Nachbarzellen wird lebendig (Fortpflanzung).

Dieses Regelwerk ist in der Methode `org.bitstorm.gameoflife.GameOfLifeGrid.next()` implementiert. Als Nachbarn gelten nicht nur die Zellen oberhalb, unterhalb, rechts, links von einer Zelle,

sondern auch die diagonal angrenzenden (8er-Nachbarschaft). Es gibt auch abweichende Regelwerke (sogenannte *Welten*), die hier jedoch nicht von Interesse sind.

Aufbau

Packages

- `org.bitstorm`: Dies ist das Oberpackage, das die ganze Applikation beinhaltet.
- `org.bitstorm.gameoflife`: In diesem Package liegt der Code zur Erzeugung der grafischen Benutzeroberfläche und die eigentliche Spiellogik.
- `org.bitstorm.util`: Dieses Package beinhaltet Hilfsklassen wie Dialoge und Utility-Klassen für allgemeine Aufgaben.

Klassen

- `org.bitstorm.gameoflife`
  - `Cell` repräsentiert eine einzelne Zelle. Diese kennt ihre Position und die Anzahl (aktiver?) Nachbarzellen.
  - `CellGrid` definiert ein Interface, das die Funktionalität eines Rasters bestimmt. Es kann der Status einer Zelle an einer bestimmten Position (mittels x/y-Koordinaten) und die Dimension des Grids abgefragt werden. Das Raster kann auch in der Grösse angepasst werden.
  - `CellGridCanvas` erweitert ein AWT-Canvas und behandelt die Interaktion des Benutzers mit dem Raster (Event-Handling). Der Benutzer kann per Mausklick den Status von Zellen ändern. Die Benutzeroberfläche ist skalierbar, die einzelnen Zellen haben jedoch eine feste Grösse: Beim Skalieren wird also nicht die Grösse der Zellen, sondern deren Anzahl angepasst. (So kann es passieren, dass Statusinformationen über Zellen am Rand nach Verkleinerung und anschliessender Vergrösserung des Fensters verlorengehen.) Die Interaktion mit dem Raster erfolgt über das `CellGrid`-Interface.
  - `GameOfLife` erweitert ein AWT-Applet und baut die ganze Benutzeroberfläche auf. Neben dem eigentlichen Spielfeld (Zellenraster) gibt es unten auch noch Steuerelemente mit vordefinierten Figuren, Schaltflächen zur Steuerung des Ablaufs (*Next*, *Start*, Auswahl der Geschwindigkeit), eine Auswahl für die Zellengrösse und eine Statusanzeige für die Generation (Nummer des aktuellen Zyklus). Die Berechnung der nächsten Generation wird in einen eigenen Thread ausgelagert, wozu das `Runnable`-Interface implementiert wird. TODO
- `org.bitstorm.util`

Log-Ereignisse

nach Log-Level

Error

- bei Exceptions

Warning

- wenn keine Änderungen mehr stattfinden

Debug

- Benutzerinteraktoin

Info

- neue Generation

Trace

- Statusänderung einzelner Zelle

Spezielles

Das Projekt enthält beispielsweise solchen Code:

```
System.out.println("Hello, world!\n");
```