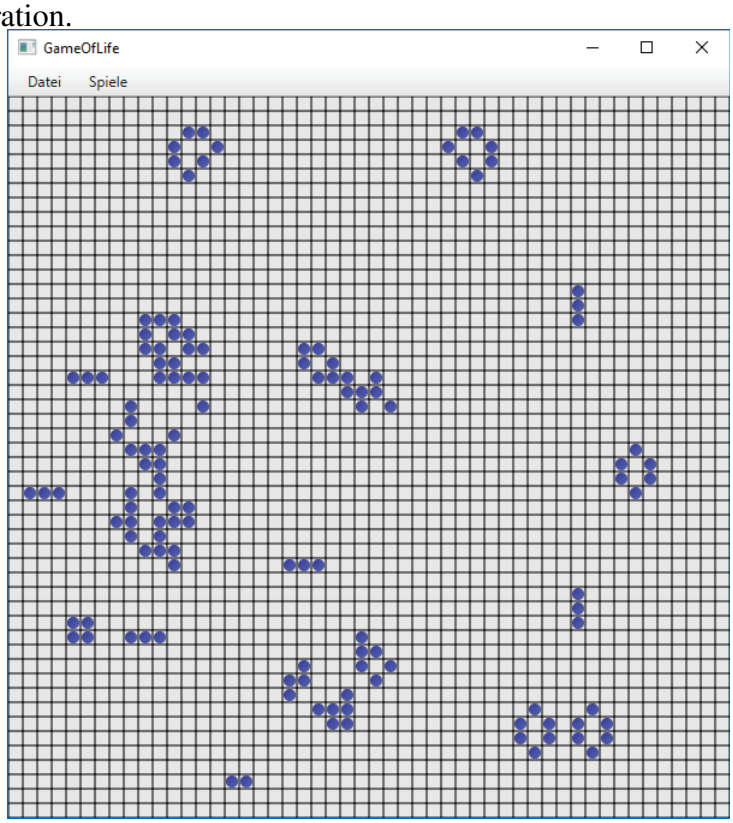
Pflichtaufgabe 2 – Game of Life

Dieses Spiel wurde erstmalig Ende der 60er Jahre von J. H. Conway vorgeschlagen und detailliert untersucht. Eine genaue Beschreibung ist unter Anderem in Wikipedia zu finden: `http://de.wikipedia.org/wiki/Game_of_Life`. Das Spiel basiert auf einem zweidimensionales Feld aus n Zeilen und m Spalten. Jede Zelle des Feldes besitzt einen der zwei Zustände „lebend“ (im Bild unten blau dargestellt) oder „tot“. Die zu Beginn lebenden Zellen bilden die Anfangsgeneration.



In jedem Zeittakt wird eine neue Generation nach den folgenden Regeln ermittelt:

- Eine lebende Zelle lebt auch in der nächsten Generation, wenn zwei oder drei ihrer acht Nachbarn in der aktuellen Generation leben.
- Eine lebende Zelle stirbt an Überbevölkerung, wenn mehr als drei ihrer acht Nachbarn in der aktuellen Generation leben.
- Eine lebende Zelle stirbt an Vereinsamung, falls weniger als zwei ihrer acht Nachbarn in der aktuellen Generation leben.

- Eine tote Zelle wird in der nächsten Generation lebendig, wenn genau drei ihrer acht Nachbarn in der aktuellen Generation leben.

Das Feld ist nicht zyklisch aufgebaut. Das heißt, dass eine Zelle am Rand des Spielfeldes weniger Nachbarzellen als eine Zelle im Zentrum besitzt.

2.1 Initialisierung

Verwenden Sie die leere Klasse `GameOfLifeLogic` aus dem Grundgerüst als Basis für die Logik des Spiels. Die Visualisierung `GameOfLifeApplication` und `GameOfLifeCanvas` sowie einige vordefinierte Startsituationen (Klasse `Games`) sind bereits vorhanden. Das Vorgabeprojekt können Sie im Ilias auf den Übungsseiten herunterladen. Denken Sie bei der Implementierung daran, dass Sie die alte und die neue Generation speichern müssen. Sie können nicht schon beim Berechnen der neuen Generation die alte überschreiben.

2.2 Betrieb

Ihre Klasse `GameOfLifeLogic` beinhaltet die Logik des Spiels. Damit Ihre Logik eingesetzt werden kann, müssen Sie die folgenden Methoden implementieren:

- `void setStartGeneration(boolean[][] generation)`: Durch den Aufruf dieser Methode durch die Visualisierung bekommen Sie die im Menü ausgewählte Startsituation als zweidimensionales Array übergeben.
- `void nextGeneration()`: Berechnet die Folgegeneration der aktuellen und trägt sie in das Array ein. Beachten Sie, dass während der Berechnung die neue Generation nicht die alte überschreiben darf. Das ist erst dann erlaubt, wenn die neue Generation vollständig berechnet wurde.
- `boolean isCellAlive(int x, int y)`: Testet, ob die Zelle an der Position (x, y) in der aktuellen Generation lebt. Dann ist der Rückgabewert `true`, ansonsten `false`.

Hinweise:

- Kommentieren Sie alle public deklarierten Member (Methoden, Konstruktoren und Attribute) Ihres Quellcode vollständig mit Javadoc (auch Getter- und Setter-Methoden).
- Prüfen Sie die Einhaltung der Code-Konventionen von Oracle: Diese müssen mittels des Plugins für Checkstyle sichergestellt werden.