Prof. Heike Wehrheim
und Mitarbeiter

Paderborn, den 21. Dezember 2007

Weihnachtsübung zur Vorlesung

Grundlagen der Programmierung 1

WS 2007/08

Bitte beachten :
Die Aufgaben auf diesem Blatt müssen nicht abgegeben werden und werden nicht korrigiert. Sie sollen dazu dienen, zusätzliche Übungsmöglichkeiten für die Zeit zwischen Weihnachten und Neujahr zu bieten. Es dürfen jedoch Fragen zu den Aufgaben an Eure Tutoren oder an Dietrich Travkin (per E-Mail) gestellt werden.

AUFGABE 1:

Im Rahmen dieser Aufgabe soll eine Simulation entwickelt werden, welche das Verhalten von speziellen Lebewesen beschreibt. Diese Lebewesen leben in einer Landschaft aus rechteckigen Feldern, Zellen genannt. Ein Lebewesen hat die Größe einer Zelle. Ausgehend von einer Anfangspopulation entwickelt sich die Gemeinschaft der Lebewesen nach folgenden Regeln weiter:

- Wenn ein Wesen weniger als zwei lebendige Nachbarn hat, stirbt es an Vereinsamung. Hat es mehr als drei lebendige Nachbarn, stirbt es aufgrund von Überbevölkerung.
- Wenn eine leere Zelle genau drei lebendige Nachbarn hat, wird in dieser Zelle ein neues Lebewesen geboren.
- Alle Geburten und Sterbefälle geschehen gleichzeitig, so dass sich die Gemeinschaft in Generationssprüngen entwickelt. Ein sterbendes Wesen kann daher bei der Geburt eines neuen in einer leeren Zelle behilflich sein, nicht aber den Tod eines Nachbarn verhindern, indem es die Bevölkerungsdichte herabsetzt.
- Lebendige Nachbarn im Sinne dieser Regeln sind alle Wesen, die in einer direkt senkrecht, waagerecht oder diagonal benachbarten Zelle leben.

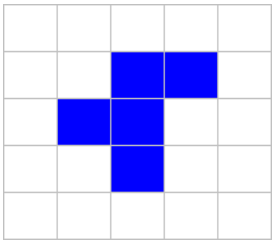

Abbildung 1: Generation 0

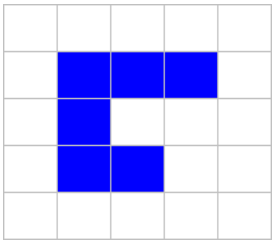

Abbildung 2: Generation 1

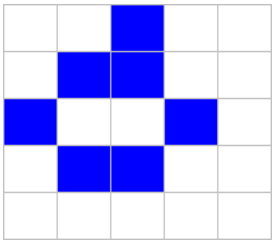

Abbildung 3: Generation 2

Die Abbildung 1 stellt ein Beispiel für eine Anfangspopulation dar. Die nächste Generation zu dieser Ausgangspopulation ist in der Abbildung 2 zu sehen und die darauf folgende in der Abbildung 3. In Abbildung 1 hat z.B. die Zelle mit den Koordinaten (2, 2) (also genau in der Mitte) insgesamt 4 belebte Nachbarzellen. Laut den oben genannten Regeln stirbt

das Lebewesen in dieser Zelle im nächsten Schritt wegen Überbevölkerung. Gleichzeitig hat die leere Zelle (1, 3) (also die zweite von links und vierte von oben) genau drei lebendige Nachbarn, was dazu führt, dass an dieser Stelle ein neues Lebewesen geboren wird (vgl. Abb. 2).

Ihre Aufgabe ist es, eine bereits implementierte grafische Oberfläche um einen Algorithmus zu erweitern, der die Entwicklung einer Population nach den oben beschriebenen Regeln simuliert.

Die zur Verfügung stehende grafische Oberfläche ist in Abbildung 4 zu sehen. Hier hat der Benutzer die Möglichkeit, durch Anklicken einzelner Zellen ihren Zustand von leer zu bevölkert und umgekehrt zu ändern und somit eine Anfangspopulation zu wählen. Der Button „Clear“ leert alle gefüllten Zellen. Eine Simulation kann schrittweise durch den Button „Next“ oder automatisch durch den Button „Start“ durchgeführt werden. Die automatisch laufende Simulation wird durch den Button „Stop“ angehalten. Die Zeit zwischen zwei Simulationsschritten (Bestimmen einer nächsten Population bzw. Generation) wird bei der automatischen Simulation durch das Feld „Update period“ in Millesekunden angegeben. Mit Hilfe der Felder „cols“ (Anzahl Spalten), „rows“ (Anzahl Zeilen) und des Buttons „Set size“ kann die Größe des Feldes (Gitters) geändert werden. Das Label „Iterations“ zeigt die Anzahl der bereits simulierten Schritte (bzw. die Nummer der aktuellen Generation) an.

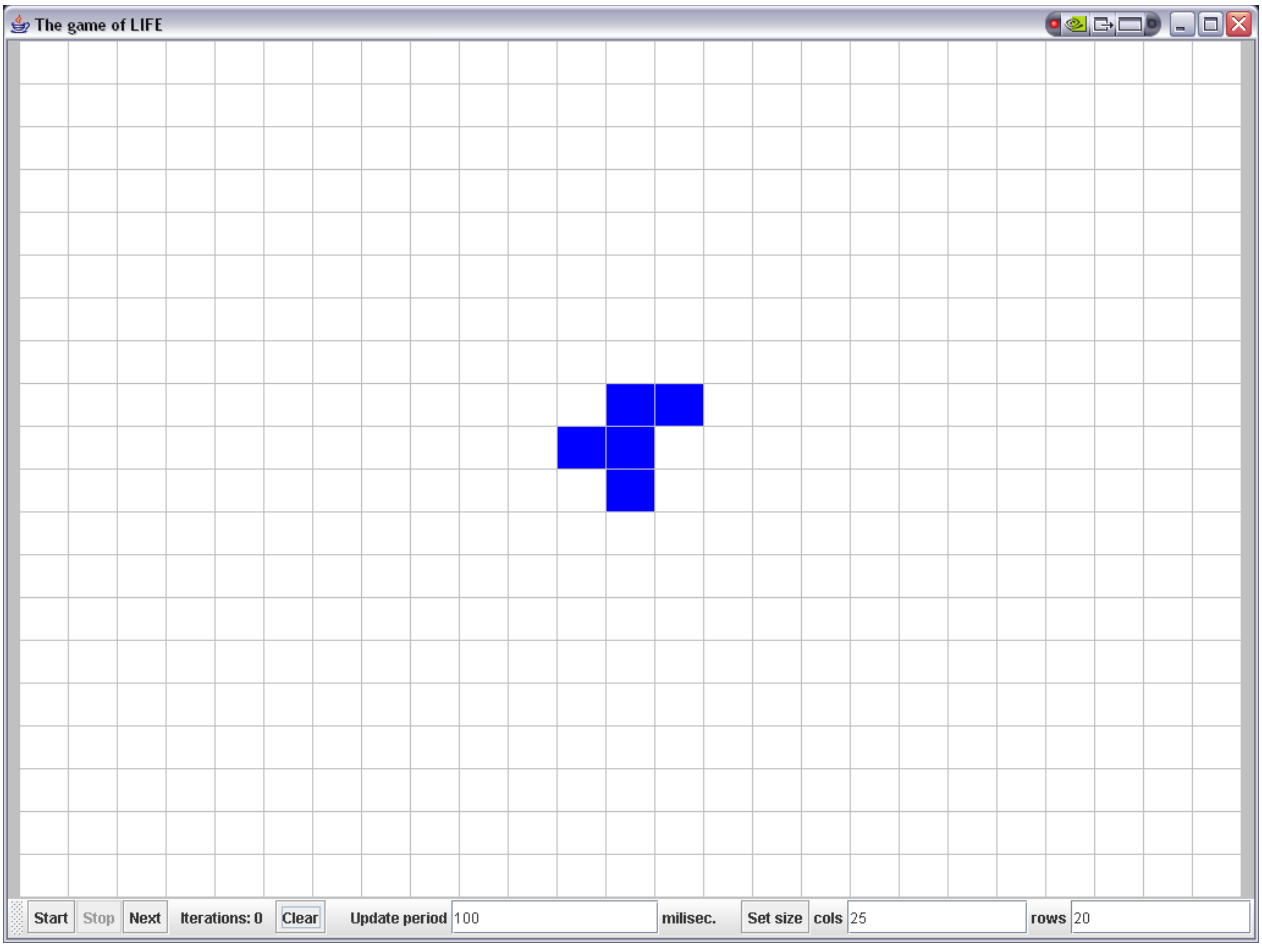

Abbildung 4: Ansicht der grafischen Oberfläche zur Simulation

Der zu implementierende Algorithmus soll durch Erweitern zweier bereits vorgegebener Klassen implementiert werden. Diese können zusammen mit den Klassen, welche die grafische Oberfläche der Simulation realisieren, von der Vorlseungsseite heruntergeladen werden. Die zu erweiternden Klassen sind `GameOfLife` und `GameOfLifeField`. Ein `GameOfLife`-Objekt repräsentiert eine Simulationsinstanz und verwendet zur Speicherung der aktuellen Zelleninhalte ein `GameOfLifeField`-Objekt.

Die bereits implementierten und zu implementierenden Methoden und Variable der Klassen sind in Abbildung 5 dargestellt. Die zu implementierenden Methoden sind durch kursive Schrift gekennzeichnet. Die gewünschte Funktionalität der Methoden ist in dem Quellcode durch javadoc-Kommentare beschrieben.

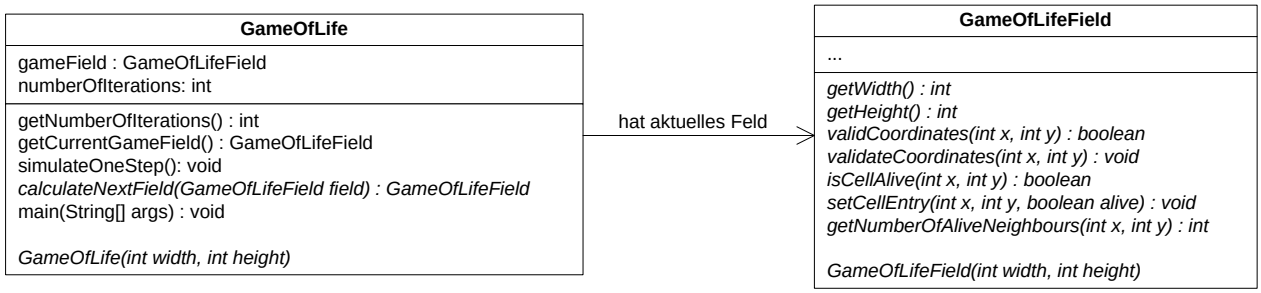

Abbildung 5: Zu erweiternde Klassen

Zur Lösung der Aufgabe gehen Sie wie folgt vor:

- Fügen Sie der Klasse `GameOfLifeField` Variable hinzu, um den aktuellen Zustand aller Zellen zu speichern und implementieren Sie alle leeren Rümpfe der in Abbildung 5 markierten Methoden der Klasse `GameOfLifeField`. Testen Sie die Klasse, indem Sie in einer `main`-Methode eine Anfangspopulation festlegen und die Funktionalität aller Methoden durch Auswerten der Rückgabewerte bzw. der gespeicherten Zelleninhalte prüfen. Testen Sie insbesondere die Methode `getNumberOfAliveNeighbours`, welche die Anzahl der lebendigen Nachbarn einer Zelle bestimmen soll.

- Implementieren Sie nun die beiden leeren Methoden der Klasse `GameOfLife` (eine davon ist der Konstruktor) und testen Sie die Methode `calculateNextField`. Diese soll zu einer gegebenen Zellenbelegung (Population) die Population der folgenden Generation bestimmen und zurückgeben. Achten Sie darauf, dass hier nicht das übergebene `GameOfLifeField`-Objekt verändert wird, sondern ein neues Objekt erstellt wird, welches die nächste Generation repräsentiert.

Nachdem Sie die beiden Klassen wie oben beschrieben erweitert und zusammen mit den vorgegebenen Klassen kompiliert haben, können Sie die grafische Oberfläche durch Ausführen der `main`-Methode der Klasse `GameOfLife` starten. Zum Vergleich können Sie sich die Simulation unter folgender URL ansehen:
http://wwwcs.uni-paderborn.de/cs/ag-schaefer/Lehre/GP/WS07/GameOfLife/GameOfLIFE.html.

Probieren Sie die in den Abbildungen 6 bis 12 dargestellten Anfangspopulationen aus. Unter dem Suchbegriff „Game of Life“ finden Sie im Internet zahlreiche weitere interessante Populationen.

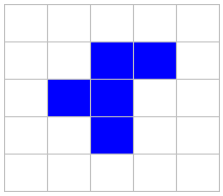

Abbildung 6: Beispiel 1

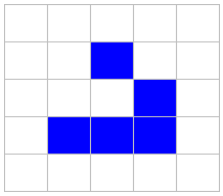

Abbildung 7: Beispiel 2

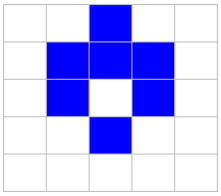

Abbildung 8: Beispiel 3

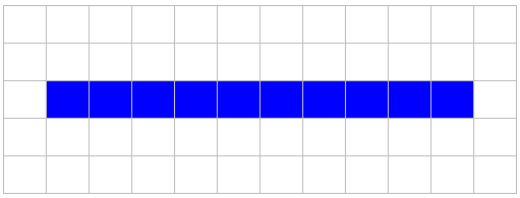

Abbildung 9: Beispiel 4

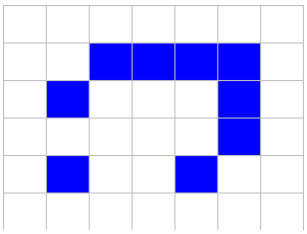

Abbildung 10: Beispiel 5

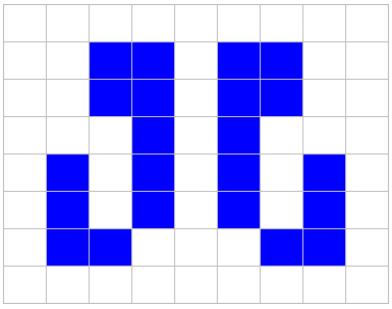

Abbildung 11: Beispiel 6

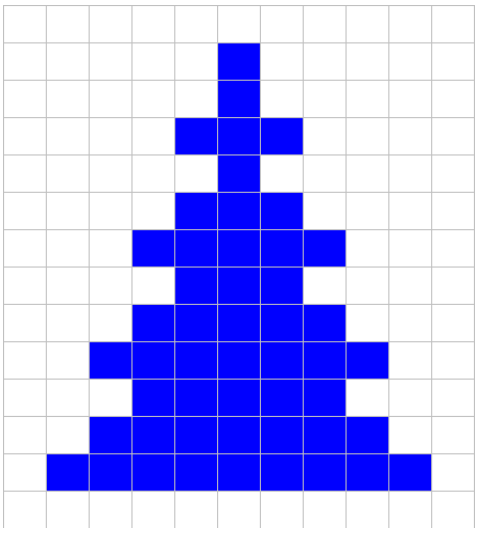

Abbildung 12: Beispiel 7

Viel Spaß und Erfolg beim Ausprobieren,
Frohe Weihnachten und guten Rutsch ins Neue Jahr!