Algorithmen und Datenstrukturen Übung 2

Auszuarbeiten bis 18.10.17

1. Grobanalyse von Sortieralgorithemen (5 + 3 Punkte)

a) Führen Sie jeweils für das Auswahlsortieren und das Einfügesortieren eine Grobanalyse durch. Es sollen sowohl die minimalen, maximalen und durchschnittlichen Schleifendurchläufe als auch Vergleiche ermittelt werden, wobei nur Vergleiche in Bezug auf Feldzugriffe berücksichtigt werden.

b) Implementieren Sie das Verfahren und versuchen Sie den Code so zu „instrumentieren" d.h. mit Hilfszählern etc. anzupassen, um Ihre theoretische Betrachtung zu belegen. Die Resultate sollen am Bildschirm ausgegeben werden.

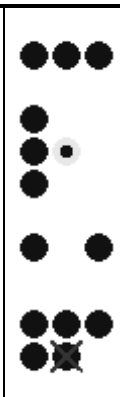
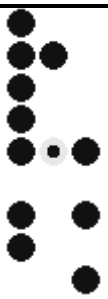
2. Game of Life (6 + 6 Punkte)

In diesem Beispiel geht es darum, so genannte zelluläre Automaten zu simulieren. Eine mögliche Definition eines zellulären Automaten ist die folgende:

„Ein Zellautomat ist ein rechteckiges Gitterraster aus regelmäßigen Quadraten («Zellen»), etwa wie ein Schachbrett. Jede Zelle kann eine Anzahl verschiedener Werte annehmen und hat eine begrenzte Zahl von Nachbarzellen, die sie beeinflussen können. Das Muster oder der «Zustand» des gesamten Rasters ändert sich in einzelnen Schritten entsprechend einer Reihe von «Übergangsregeln», die gleichzeitig für jede Zelle gelten."
(Lebensnetz, von Fritjof Capra, 1996, Seite 223)

Zweidimensionale zelluläre Automaten werden auch „*Game of Life*" genannt. Das „*Game of Life*" ist eigentlich ein Spezialfall eines Räuber-Beute-Modells. Im Räuber-Beute-Modell hat man zwei Populationen (Räuber und Beute), die sich gegenseitig limitieren.

Im „*Game of Life*" hingegen gibt es nur eine einzige Population. Ob eine Population zu- oder abnimmt hängt davon ab, ob ein Individuum eine ideale Anzahl Nachbarn hat. Bei zu vielen Nachbarn machen sie sich gegenseitig das Futter streitig und verhungern. Bei zu wenig Nachbarn gibt es Probleme mit der Vermehrung. Bei welcher Anordnung von Nachbarn eine aktive Zelle weiterlebt, abstirbt oder neu entsteht, wird mit Regeln beschrieben.
Beim zweidimensionalen „*Game of Life*" werden 8 Nachbarn berücksichtigt:

Hat eine Zelle 2 oder 3 Nachbarn, dann stimmt das soziale Umfeld– sie überlebt Hat eine *'unbelebte'* Zelle drei Nachbarn, so entsteht eine neue Zelle Hat eine Zelle nur einen oder keine Nachbarn, stirbt sie an Vereinsamung Hat sie mehr als 3 Nachbarn, so ist nicht ausreichend Nahrung vorhanden - sie stirbt auf Grund Überbevölkerung			

Die "Welt" der Zellen kann am Rande aufhören oder sich zu einer runden Welt oder einem Torus krümmen.

a) Design

Erstellen Sie ein Design für eine Simulationsumgebung für das *Game of Life*. Denken Sie dabei an die Verteilung der Verantwortung. Trennen Sie Eingabe bzw. Ausgabe von der Spiellogik - eventuell „möchten" Sie in Zukunft eine grafische Benutzeroberfläche für Ihr *Game of Life* entwickeln.

Identifizieren Sie mögliche Klassen und deren Beziehungen zueinander anhand der obigen allgemeinen Beschreibung, denken Sie auch über die Eigenschaften der Klassen nach und realisieren Sie ein entsprechendes `UML-Klassendiagramm`.

Beachten Sie die folgenden Anforderungen:

- Der Simulator muss auf jeden Fall die folgende Funktionalität anbieten:
  - Erzeugen einer neuen Welt, bei der die Zellen nach dem Zufallsprinzip im Lebensraum verteilt werden.
  - Vorrückung der Zeitscheibe um eine Zeiteinheit `t = t + 1`
  - Vorrückung der Zeitscheibe um *n* Zeitscheiben `t = t + n`
- Der Lebensraum hat eine einstellbare Größe
  - Darstellung des aktuellen Lebensraumes (Achten Sie hierbei besonders auf die sinnvolle Anwendung von Delegation)
- Die „Form" der Welt ist wählbar:
  - Welt endet am Rand
  - Welt ist ein Torus
- Jede Zelle verfügt über Eigenschaften wie z.B. Alter, Farbe etc.

Beschreiben Sie mit Hilfe eines `Sequenzdiagramms` den Ablauf während der Vorrückung der Zeitscheibe um eine Zeiteinheit.

b) Implementierung

Implementieren Sie Ihr Design unter der Verwendung von Interfaces und abstrakten Klassen

Die Ausgabe des jeweiligen Spielzustandes erfolgt vorerst mittels `System.out`. Bedenken Sie allerdings, dass demnächst vielleicht eine grafische Darstellungsform realisiert werden muss.

Überlegen Sie sich eine sinnvolle textuelle Darstellungsform für die Welt, die Zellen und auftretende Ereignisse. Der Testtreiber muss die folgenden Testvarianten unterstützen:

- Weiterbewegung der Zeitscheibe um eine Zeiteinheit
- Weiterbewegung der Zeitscheibe um *n* Zeiteinheiten
- Auslösen von Naturereignissen

Die Initialisierung der Welt d.h. Zuteilung von Zellenverhalten zu bestimmten Zellen, Festlegung des Intervalls von Naturereignissen, etc. kann hart-kodiert erfolgen. Wichtig ist, dass möglichst alle Verhaltenskombinationen getestet werden.