

Übungen zu Informatik B

Sommersemester 2004

Blatt 5

Hinweis: Der Abgabetermin für dieses Aufgabenblatt ist erst in **zwei Wochen**, da am Donnerstag der kommenden Woche ein Feiertag ist. Es wird in der nächsten Woche also kein neues Aufgabenblatt geben, die Übung findet aber trotzdem statt.

Aufgabe 1 (4 Punkte)

Erläutern Sie den Unterschied zwischen den beiden in der Vorlesung vorgestellten Interfaces *Collection* und *Map*. Was versteht man unter dem Begriff *View* auf eine *Map*?

Erklären Sie das *Visitor* Pattern. Wie wird ein *Visitor* verwendet, und worin besteht der Unterschied zu einem *Iterator*?

Aufgabe 2 (6 Punkte)

Implementieren Sie eine *sortierte Collection* vom Typ `SimpleCollection`, die ihre Einträge in Form eines binären Baums speichert. Verwenden Sie dazu die in der Vorlesung gezeigte Realisierung eines Suchbaums in der Klasse `MyTreeMap`.

Für den *Iterator* (der natürlich ebenfalls implementiert werden muß), soll das erweiterte Interface *SimpleIterator* vom letzten Aufgabenblatt verwendet werden, d.h. es soll auch eine `remove()` Operation für den Baum geben. Hierfür muß zunächst die Klasse `MyTreeMap` um die *Iterator-Funktionalität* erweitert werden.

Passen Sie Ihr Testprogramm aus Aufgabe 5 von Blatt 4 so an, daß damit diese *Collection-Implementierung* getestet werden kann.

Aufgabe 3 (6 Punkte)

Implementieren Sie *Conway's Game of Life*:

In einer Matrix leben Kolonien von Lebewesen. Die Lebensqualität hängt stark von der Anzahl der angrenzenden Kolonien ab. Wird eine Mindestanzahl von Nachbarn (typischerweise 2) unterschritten, so stirbt die Kolonie in der nächsten Generation an Einsamkeit. Wird eine Maximalanzahl (typischerweise 3) überschritten, so stirbt die Kolonie, weil nicht mehr genug Nahrung zur Verfügung steht. Hat ein *leeres Feld* diese Maximalanzahl von Nachbarn, so wird in diesem Feld eine neue Kolonie gegründet.

Geben Sie sich für beide Werte sowie die Spielfeldgröße feste Grenzen vor (z.B. 20×20), und implementieren sie das Spiel zunächst in einfacher Form mit textueller Ausgabe im Terminal.

Verwenden Sie zur Repräsentierung des Spielfelds eine eigene Klasse, die alle notwendigen Informationen über den aktuellen Zustand der Kolonien speichert und daraus die Nachfolgeneration berechnen kann. Ihre Klasse sollte mindestens folgende Methoden enthalten:

— bitte wenden —

- eine *Eingabemethode*.

Da die echte Eingabe einer Startsituation etwas langweilig ist, wollen wir die Matrix zufällig füllen. Benutzen Sie hierfür die Klasse *java.util.Random*, und überlegen sie sich einen Algorithmus zum Füllen der Ausgangsmatrix mit Kolonien.

- eine Methode für den *Generationswechsel*.

Diese betrachtet jeweils alle angrenzenden Matrixelemente für ein besetztes bzw. leeres Feld und entscheidet über die Existenz einer Kolonie in der folgenden Generation. Achtung: Der Generationswechsel passiert – anschaulich formuliert – „gleichzeitig“ für alle Kolonien, d.h. die Veränderungen betreffen nicht die Berechnung des Folgezustands für andere Kolonien der gleichen Generation.

Versuchen Sie bei der Implementierung darauf zu achten, daß die Repräsentierung des Spielfelds von der Art der Darstellung unabhängig ist, d.h. es sollte auch möglich sein, diese Klasse für eine ganz andere Art der Ausgabe wiederzuverwenden. Definieren Sie sich dazu ein eigenes *Interface*, das festlegt, welche Operationen das Spielfeld nach außen zur Verfügung stellt (das können mehr als die beiden oben angegebenen sein), und lassen Sie Ihre Klasse dieses Interface implementieren.

Die Steuerung des Programmablaufs (Spielfeld füllen, aktuellen Zustand des Feldes anzeigen, neue Generation berechnen lassen usw.) soll in einer separaten Klasse realisiert werden. Geben Sie für eine existente Kolonie hierbei ein einzelnes Zeichen, sonst ein Leerzeichen aus.

Hinweis: Um zwischen der Darstellung einer Generation und der Berechnung der Folgegeneration das Programm für eine bestimmte Zeit warten zu lassen, kann die Klassenmethode *sleep()* aus der Klasse *java.lang.Thread* verwendet werden.

Aufgabe 4 (4 Punkte)

Implementieren Sie nun eine *grafische Oberfläche* für Ihr Programm aus Aufgabe 2 unter Verwendung von grafischen Elementen aus dem *AWT*. Die Darstellung soll also in einem Fenster erfolgen, in dem der jeweils aktuelle Zustand des Spielfelds angezeigt wird. Zur Anzeige einer einzelnen Kolonie kann z.B. ein *Label* verwendet werden, in dem bei Anwesenheit einer Kolonie ein Zeichen dargestellt oder alternativ eine andere Hintergrundfarbe gesetzt wird.

Neben der Darstellung der Kolonien soll das Fenster noch jeweils eine Schaltfläche zum Anstoßen eines neuen Generationswechsels und zum Beenden des Programms besitzen.

Verwenden Sie zur Gestaltung des Fensters einen (bzw. mehrere) geeignete *Layout Manager*.

Gesamtpunkte: 20