Fachbereich Informatik und Kommunikation
Prof. Dr. Marcel Luis

Westfälische Hochschule

21. März 2012

Aufgabenblatt 1 zu Objektorientierte Programmierung

Aufgabe 1.1 (Aufwand ca. 10 Stunden, Abgabe bis 06.04.2012 um 21 Uhr)

In dieser Aufgabe sollen Sie einen Algorithmus entwickeln und realisieren, um zufällige Labyrinthe zu konstruieren (so zufällig, wie es unter Verwendung der Zufallsmethode `Math.random` eben möglich ist). Die Wände und Gänge des Labyrinths sollen sich an einem Gitterraster orientieren. Die Wände liegen auf den Linien des Rasters, die Gänge auf den Kästchen des Rasters. Die konstruierten Labyrinthe sollen folgende Eigenschaften besitzen:

- Das Labyrinth hat eine rechteckige Grundform. Es ist außen von einer geschlossenen Wand umgeben.
- Im Labyrinth gibt es zwischen je zwei Rasterfeldern genau einen Weg. Dies bedeutet insbesondere, dass es – abgesehen von der Außenwand – keinen vollständig von einer Wand umschlossenen Bereich gibt.
- Jede Wand des Labyrinths führt zur Außenwand des Labyrinths.
- Jeder Gang des Labyrinths hat die Breite eines Rasterkästchens.

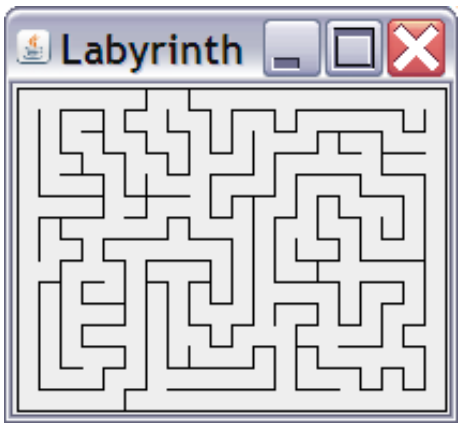

Abbildung 1: Beispiel für ein Labyrinth mit den genannten Eigenschaften. Die Breite des Labyrinths beträgt 20 Rastereinheiten, die Höhe 15 Einheiten.

Realisieren Sie im Paket `labyrinth` eine Klasse `Labyrinth` mit folgenden Methoden:

- Einen Konstruktor `Labyrinth(int, int)` zur Erzeugung eines zufälligen Labyrinths mit den oben beschriebenen Eigenschaften. Der erste Parameter gibt die Breite des Labyrinths in Rasterkästchen an, der zweite die Höhe.

Der zeitliche Aufwand dieser Konstruktormethode soll höchstens linear von der Grundfläche des Labyrinths abhängen. (In THI werden Sie bald lernen, dass man dies so formulieren kann: Der zeitliche Aufwand soll aus $O(\text{Länge} \cdot \text{Breite})$ sein.)

Für diese Anforderung ist es wichtig zu analysieren, wie häufig die Schleifen in Ihrem Programm durchlaufen werden. Wenn Sie beispielsweise zu dem Ergebnis kommen, dass es höchstens $c \cdot \mathit{Länge} \cdot \mathit{Breite}$ Schleifendurchläufe gibt (wobei c eine beliebige Konstante ist), dann ist die Zeitbedingung erfüllt.

- Eine Methode `gibHoehe()`, die die Höhe dieses Labyrinths liefert.
- Eine Methode `gibBreite()`, die die Breite dieses Labyrinths liefert.
- Eine Methode `int gibAnzahlWandelemente()`, die die Anzahl der Wandelemente dieses Labyrinths zurückgibt. Ein Wandelement ist ein Stück einer Wand des Labyrinths und hat die Länge eines Rasterkästchens. Beispiel: Die obere Außenwand des Labyrinths besteht aus so vielen Wandelementen, wie das Labyrinth breit ist.
- Eine Methode `Punkt gibStartpunkt(int)`, die den Startpunkt des Wandelements mit dem angegebenen Index liefert. Das Koordinatensystem der Punkte beginnt links oben. Der Punkt $(x = 0, y = 0)$ entspricht der linken oberen Ecke des Labyrinths, der Punkt $(x = \text{Breite}, y = \text{Höhe})$ der rechten unteren Ecke. Auf die Reihenfolge der Wandelemente kommt es nicht an. Der kleinste gültige Index ist `0`, der größte hängt von der Anzahl der Wandelemente des Labyrinths ab.
- Eine Methode `Punkt gibEndpunkt(int)`, die den Endpunkt des Wandelements mit dem angegebenen Index liefert.

Die Klasse `util.Punkt`, die Sie für diese Aufgabe benötigen, ist Ihnen im Quellcode vorgegeben und kann einfach durch Kopieren in Ihr Projekt (Paket `util` beachten) eingebunden werden. Ebenfalls vorgegeben sind Klassen `labyrinth.UILabyrinth` und `labyrinth.Labyrinthdarstellung`, die eine grafische Anwendung zum Testen und „Spielen“ realisieren. Der Start der Anwendung erfolgt durch Ausführen der Klasse `UILabyrinth`. Die Klasse enthält Konstanten für die Höhe und Breite des Labyrinths sowie für die Anzahl Pixel, die einem Rasterkästchen entspricht.

Hinweise zum Algorithmus

Die vielen Eigenschaften der Labyrinthe könnten einen vermuten lassen, der Algorithmus zu ihrer Konstruktion wäre schwierig. Tatsächlich ist er recht einfach, sodass der Anspruch dieser Aufgabe vor allem darin liegt, die Daten des Labyrinths so zu organisieren und auf ihnen so zu operieren, dass die geforderte Laufzeitbedingung erfüllt wird.

Die Aufgabe des Algorithmus ist vor allem, die Zwischenwände des Labyrinths zu konstruieren. Die äußeren Wände bilden ein Rechteck der Höhe und Breite des Labyrinths. Alle geforderten

Eigenschaften des Labyrinths lassen sich erfüllen, wenn man folgende Konstruktionsregeln beachtet.

- Man konstruiert zunächst die Außenwände.
- Neue Wandelemente werden dem Labyrinth so hinzugefügt, dass sie an *genau einer Stelle* ein schon vorhandenes Wandelement berühren. Machen Sie sich klar, was passiert, wenn Sie zulassen, dass das neue Wandelement entweder *kein* Wandelement oder *zwei* Elemente berührt.
- Man wendet die vorherige Regel solange an, wie sich ein solches neues Element noch platzieren lässt. Machen Sie sich klar, was passiert, wenn Sie mit der Konstruktion vorher aufhören.

Hinweise

- Verwenden Sie nur den Stoff der Vorlesung EPR und OPR bis Kapitel 3.
- Wenn Sie zusätzliche Klassen realisieren, legen Sie diese in das Paket `labyrinth`.
- Denken Sie an die ausreichende Dokumentation und Kommentierung Ihrer Lösung. Beachten Sie die unterschiedliche Bedeutung der *externen Dokumentation* `/** ... */` *vor* einer Klasse oder Methode und des *Implementierungskommentars* `/* ... */` *innerhalb* einer Methode. Die externe Dokumentation sagt, *was* eine Klasse oder Methode leistet, der Implementierungskommentar hilft zu verstehen, *wie* es gemacht wird. Verwenden Sie Implementierungskommentare vor allem, um den Berechnungsablauf verständlich zu machen.
- Erzeugen Sie die HTML-Dokumentation Ihrer Klasse und überzeugen Sie sich, ob Ihre externe Dokumentation sinnvoll und ohne Kenntnis des Quellcodes der Klasse hilfreich ist.
- Erstellen Sie ein `zip`-Archiv des Quellordners `labyrinth` (darin dürfen auch die beiden vorgegebenen Klassen dieses Pakets sein) und laden Sie nur diese Datei `labyrinth.zip` zu Moodle hoch.
- Im Praktikumskalender finden Sie die Termine, an denen diese Lösung im Praktikum besprochen wird. Bringen Sie zu diesen Terminen bitte die Auswertung Ihrer Lösung mit, entweder ausgedruckt oder unmittelbar auf Ihrem Rechner verfügbar.