UNIVERSITÄT BASEL

University of Basel

Dozenten
Prof. Dr. Thomas Vetter
Dr. Marcel Lüthi
Departement
Mathematik und Informatik
Spiegelgasse 1
CH – 4051 Basel

Assistent
Dr. Marcel Lüthi
Departement
Mathematik und Informatik
Spiegelgasse 1
CH – 4051 Basel

Tutoren / Tutorinnen
Claudia Grundke
Viktor Gsteiger
Simon Dold
Timo Steinebrunner
Alexander Rovner
Nikodem Kernbach
Lukas Stöckli

Erweiterte Grundlagen der Programmierung (45398-01)

Blatt 6 [10 Punkte]

Vorbesprechung	28. Oktober - 02. Oktober
Abgabe	8. November

Allgemeine Hinweise

- Wir empfehlen Ihnen, dass Sie im Buch "Sprechen Sie Java" bis und mit Kapitel 12 lesen.

Voraussetzung

- Es gelten dieselben Voraussetzungen wie für Übungsblatt 1. Wenn Sie sich betreffend der Umgebung oder der automatisierten Tests noch unsicher sind, lesen Sie bitte nochmals sorgfältig das Infoblatt oder fragen Sie die Tutoren.
- Die Zip-Datei, die auch dieses Übungsblatt enthält, muss entpackt werden. Es enthält die gesamte Übungsumgebung inklusive der automatisierten Tests. Schreiben Sie Ihre Lösungen in die dafür vorgesehenen Dateien, wie in der jeweiligen Übungsaufgabe angegeben.

Empfohlenes Vorgehen

- Wechseln Sie in den Ordner `src/main/java`. Dort finden Sie die Dateien, in welche Sie Ihren Java Code schreiben.
- Schreiben Sie Ihr Programm, kompilieren Sie dieses mit dem Java Compiler `javac` und führen Sie es mit `java` aus, wie es in der Vorlesung gezeigt wurde.
- Wenn Sie denken, dass alles in Ordnung ist, wechseln Sie zurück ins Übungsverzeichnis `uebung6` und führen da `gradlew test` aus, um zu überprüfen, ob Ihre Lösung die automatisierten Tests besteht. Überprüfen Sie auch Ihren Codestil mit `gradlew checkstyleMain`. Falls Ihr Code den Vorgaben entspricht, erhalten Sie einen Bonuspunkt.

Abgabe

Ergänzen Sie die Datei email.txt mit Ihrer Unibas E-Mail Adresse. Erstellen Sie eine Zip-Datei der gesamten Übungsumgebung (also des Verzeichnisses `uebung6`) und laden Sie dieses auf Courses (https://courses.cs.unibas.ch/) hoch.

Aufgabe 1 - Rekursion [2 Punkte]

- Im Verzeichnis `src/main/java` finden Sie die Klasse `Snowflake.png` Implementieren Sie eine Methode `drawKochCurve`. Diese Methode soll *rekursiv* (d.h. durch Aufrufe von sich selbst) folgende Kurven erzeugen:

Rekursionstiefe 1

Rekursionstiefe 2

Rekursionstiefe 3

Die Rekursionstiefe wird durch das Konstruktorargument `depth` bestimmt.
Die Distanz zwischen Start und Endpunkt der Kurve soll unabhängig von der Rekursionstiefe immer gleich sein und der Konstante `LENGTH_OF_CURVE` entsprechen.
Nutzen Sie die Methode `setStartPosition` um das Turtle **vor dem Aufruf der rekursiven Methode** auf der Zeichenfläche zu positionieren.
Zum Kompilieren der Programme müssen Sie die Turtle Bibliothek wie folgt mit angeben:

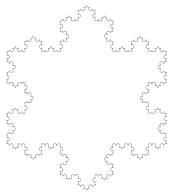
> `javac -cp .;jturtle-0.5.jar Snowflake.java` (Windows)

> `javac -cp .:jturtle-0.5.jar Snowflake.java` (Linux und MacOS)

[1 $\frac{1}{2}$ Punkt]

- Implementieren Sie dann die Methode `drawSnowflake`, welche durch mehrmaliges Aufrufen von `drawKochCurve` eine Schneeflocke zeichnet.

[$\frac{1}{2}$ Punkt]



Aufgabe 2 - Prioritätenschlange [4 Punkte]

In dieser Aufgabe implementieren Sie eine neue Datenstruktur, genannt *PriorityQueue*. Diese Datenstruktur basiert auf der in der Vorlesung besprochenen Datenstruktur *Queue* jedoch hängt die Reihenfolge von einer Priorität ab, die jedem Element zugeordnet ist.

Implementieren sie die fehlenden Methoden der Klasse `PriorityQueue`, welche Sie im Verzeichnis `src/main/java/` finden.

`Tipp:` Schreiben Sie eigene kleine Testprogramme in der Main-methode um Ihre Implementation zu testen. Wenn alles funktioniert nutzten Sie auch die mitgelieferten Tests.

Aufgabe 3 - Evolution [4 Punkte]

Wir wollen in dieser Aufgabe eine Art Evolution simulieren. Dies anhand eines vereinfachten Models, welches nicht den Anspruch erhebt, im biologischen Sinne korrekt zu sein. Wir nehmen an, dass jedes Lebewesen durch eine Gen-Sequenz beschrieben werden kann. Eine Gen-Sequenz soll eine Folge der Buchstaben 'A', 'C', 'G' und 'T' sein. Diese Gen-Sequenzen verändern sich mit der Zeit. Dies wollen wir nun simulieren. In der Aufgabe müssen Sie dazu zwei Klassen vervollständigen.

Vorbereitung

- Im Zip-Archiv, finden Sie im Ordner *src/main/java* die Klassen *Genom GenPool TestGeneticEvolution*
- Lesen Sie den Code in der Klasse `TestGeneticEvolution` und versuchen Sie zu verstehen was passiert.
- Kompilieren Sie die Klasse `TestGeneticEvolution` und führen Sie diese aus. Sehen Sie sich den Output an.
- Implementieren Sie die fehlenden Methoden in der Klasse `Genom` und schreiben Sie eigene kleine Testprogramme in der Main-methode um die Klasse zu testen.
  Tipp: Nutzen Sie die Hilfsklasse Random die Sie im Verzeichnis `src/main/java` finden, um zufällige Zahlen im gewünschten Interval zu generieren.
- Implementieren Sie die fehlenden Methoden in der Klasse `GenePool` und schreiben Sie eigene kleine Testprogramme in der Main-methode um die Klasse zu testen.
- Nutzen Sie auch die mitgelieferten Tests um die Klassen zu testen. Die Tests starten Sie wie immer mit `gradlew test`.
- Wenn alles funktioniert, kompilieren Sie die Klasse `TestGeneticEvolution` und experimentieren Sie mit den Parametern der Simulation.