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 7 [10 Punkte]

Vorbesprechung	4. November - 8. November
Abgabe	15. November

Allgemeine Hinweise

- Wir empfehlen Ihnen, dass Sie im Buch "Sprechen Sie Java" bis und mit Kapitel 11 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

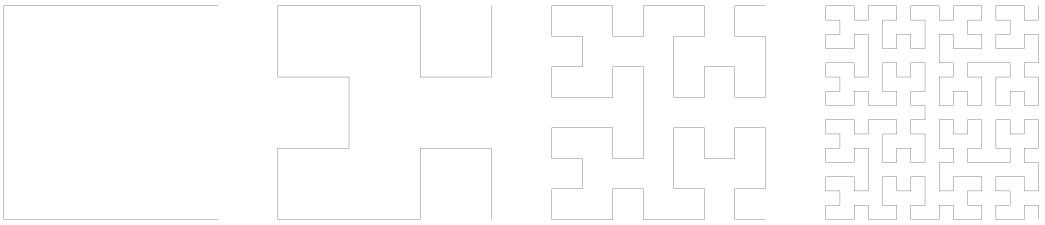
- Wechseln Sie in den Ordner `src/main/java`. Dort finden Sie die Dateien, in welche Sie Ihren Java Code schreiben.
- Schreiben Sie Ihr Programm, kompilieren Sie dieses mit dem Java Compiler `javac` und führen Sie es mit `java` aus, wie es in der Vorlesung gezeigt wurde.
- Wenn Sie denken, dass alles in Ordnung ist, wechseln Sie zurück ins Übungsverzeichnis `uebung7` 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 `uebung7`) und laden Sie dieses auf Courses (https://courses.cs.unibas.ch/) hoch.

Aufgabe 1 - Hilbertwalk [3 Punkte]

Sie finden im Verzeichnis `src/main/java` die Klasse `Hilbertwalk`. Implementieren Sie die Methode `drawHilbert`, welche die Hilbertkurve zeichnet. Die Hilbertkurve ist eine rekursiv definierte Kurve, die auf den ersten vier Rekursionstiefen wie folgt aussieht:



Rekursionstiefe 1 Rekursionstiefe 2 Rekursionstiefe 3 Rekursionstiefe 4

Um die Kurve zu zeichnen, müssen Sie zwei rekursive Methoden, `hilbertA` und `hilbertB` definieren, welche sich gegenseitig aufrufen. Die Regeln sind wie folgt. Sie starten mit einer Methode `hilbertA`, welche mittels Turtle folgendes Sequenz definiert:

$$\text{methodeA:} -BF+AFA+FB-$$

Das $+$ respetive $-$ bedeutet, dass sie 90 Grad nach rechts, respektive links drehen. Ein F bedeutet, dass Sie das Turtle nach vorne bewegen müssen. Ein B bedeutet, dass Sie die Methode `hilbertB` aufrufen. Die Methode `hilbertB` wird nach dem ähnlichen Muster definiert:

$$\text{methodeB:} +AF-BFB-FA+$$

Ein A bedeutet hier, dass Sie wiederum die Methode `hilbertA` aufrufen müssen.

Beginnen Sie die Aufgabe, indem Sie jede Linie mit der fixen Länge zeichnen, die in dem Feld `lengthLineSegment` definiert ist. Die Zeichnung wird dadurch auf jeder Rekursionsebene grösser. Passen Sie in einem zweiten Schritt die Methode `computeLengthOfLineSegment` an, so dass die Seitenlänge jeder Zeichnung der Konstanten `BASE_LENGTH` entspricht.
Tipp: Sie brauchen dafür die Methode `Math.pow(a, b)` welche a^b berechnet.

Zum Kompilieren der Programme müssen Sie die Turtle Bibliothek wie folgt mit angeben:

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

Entsprechendes gilt beim Ausführen der Programme.

Aufgabe 2 - Komplexe Zahlen [2 Punkte]

Eine komplexe Zahl ist eine Zahl $a+bi$ mit folgenden Rechenregeln:

$$\text{Addition:} \quad (a+bi)+(c+di)=(a+c)+(b+d)i \tag{1}$$

$$\text{Multiplikation:} \quad (a+bi)(c+di)=(ac-bd)+(ad+bc)i \tag{2}$$

$$\text{Betrag:} \quad |(a+bi)|=\sqrt{a^2+b^2} \tag{3}$$

Im Verzeichnis `src/main/java` finden Sie die Klasse `Complex`. Implementieren Sie die fehlenden Methoden.

Beachten Sie, dass Sie die Operationen Addition und Multiplikation jeweils in zwei Varianten implementieren müssen. In der einen Variante soll der berechnete Wert als neues Objekt zurückgegeben werden. In der zweiten Variante wird das aktuelle Objekt verändert.

Testen Sie Ihre Klasse mit den mitgelieferten Tests.

Aufgabe 3 - Mandelbrotmenge [2 Punkte]

In dieser Aufgabe schreiben Sie ein Programm, dass die Mandelbrotmenge darstellt. Die Mandelbrotmenge ist definiert als die Teilmenge der komplexen Zahlen c, für die die Folge $z_0 = 0, z_{n+1} = z_n^2 + c$ beschränkt ist. D.h., die Mandelbrotmenge besteht aus den Zahlen, für die es eine Konstante k gibt, so dass alle Elemente der Folge $z_n(c)$ kleiner k sind.

$$\{c \in \mathbb{C} \mid \exists k \forall n : z_n < k \text{ mit } z_0 = 0, z_{n+1} = z_n^2 + c\} \tag{4}$$

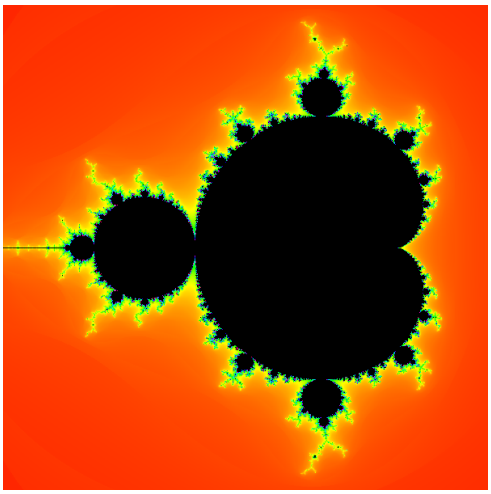
Man kann diese Menge darstellen, indem man die zu ihr gehörenden Punkte der komplexen Zahlenebene einfärbt. Um die obige Definition für eine gegebene Zahl c zu testen, müsste man *alle* Elemente der Folge z_n betrachten, was in der Praxis natürlich nicht möglich ist. Darum stellen wir stattdessen die "Fluchtgeschwindigkeit" der Folge $z_n(c)$ dar. Die "Fluchtgeschwindigkeit" definieren wir als das kleinste n von $z_n(c)$, so dass $|z_n(c)| > 2$.

Im Verzeichnis `src/main/java` finden Sie die Klasse `Mandelbrot`. Implementieren Sie die Methode `computeMandelbrot`, welche für eine gegebene Zahl die Mandelbrot Folge berechnet und ein Resultatobjekt zurückgibt, welches die Resultate der Berechnung enthält.

Testen Sie Ihr Programm mit den mitgelieferten Tests.

Aufgabe 4 - Visualisierung der Mandelbrotmenge [2 Punkte]
In dieser Aufgabe visualisieren Sie die Mandelbrotmenge. Dafür implementieren Sie die Methode `createMandelbrotVisualization` . Dafür laufen Sie über die Pixel eines Bildes, und ordnen jedem Pixel eine komplexe Zahl mit der Methode `pixelPosToComplexNumber`. Berechnen Sie dann die Mandelbrotfolge mit der in der vorigen Aufgabe implementierten Methode und nutzen Sie die Hilfsklasse `ColorPalette` um eine Farbe entsprechend der Berechnung auszuwählen.

Wenn Sie das Programm kompilieren und mit den in der Main-Methode angegebenen Parametern ausführen, sollten Sie etwa folgendes Bild erhalten.



Aufgabe 5 - Color Klasse [1 Punkte]

Implementieren Sie die Methode `ColorPalette.colorToGreyscale`. Diese soll eine Farbe, wie die von `ColorPalette.getColor` zurückgegeben wird, in Graustufen umwandeln. Dies schaffen Sie, indem die R G B Komponente der Farbe jeweils durch das gewichtete Mittel $R * 0.2989 + G * 0.5870 + B * 0.1140$ ersetzt wird.

Lesen Sie die Dokumentation der Klasse `Color` um die richtigen Methoden der Klasse `Color` zu finden: `https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/java/awt/Color.html`.

Testen Sie Ihre Implementation mit den mitgelieferten Tests.

Visualisieren Sie nun die Mandelbrotmenge auch als Graustufenbild.