Grundlagen der Programmierung

Marcel Lüthi
Andreas Morel-Forster
HS 22

Universität Basel
Fachbereich Informatik

Übung 9

Voraussetzung

- Ein JDK ist installiert.
- Installierte IDE, Visual Studio Code sowie die Plugins für Java und Gradle
- Wenn Sie die Vorlesung verpasst haben, dann empfehlen wir Ihnen die Unterlagen anzuschauen.
- Die Zip-Datei, die auch dieses Übungsblatt enthält, muss entpackt werden. Es enthält die gesamte Übungsumgebung. Schreiben Sie ihre Lösungen in die dafür vorgesehenen Dateien, wie in der jeweiligen Übungsaufgabe angegeben.

Wichtiger Hinweis

- *Achten Sie auf guten Programmierstil.*

Aufgabe 9.1 (Verkettete Listen)

Sie finden im Verzeichnis `src/main/java` die Klasse `LinkedList`. Ergänzen Sie die fehlenden Methode `max`, `reorder` und `invert`.

- Die Methode `max` soll das Element mit dem grössten Wert in der Liste suchen und diesen zurückgeben. Wenn die Liste leer ist geben Sie die kleinstmögliche Integerzahl (`Integer.MIN_VALUE`) zurück.
- Die Methode `reorder` soll die Liste von Zahlen so umsortieren, dass alle negativen Zahlen vor den positiven Zahlen stehen. Die Reihenfolge innerhalb der positiven und negativen Zahlen ist beliebig. Die Zahl 0 können Sie für diese Aufgabe als positiv betrachten.
- Die Methode `invert` soll die Reihenfolge der Elemente umkehren.

Die letzten zwei Aufgaben müssen durch geschicktes Verändern der *Verlinkung* der Knoten gelöst werden. Es dürfen keine zusätzlichen Datenstrukturen (wie zum Beispiel ein Stack oder eine weitere verkettete Liste) verwendet werden.

Aufgabe 9.2 (Komplexe Zahlen)

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 9.3 (Mandelbrotmenge)

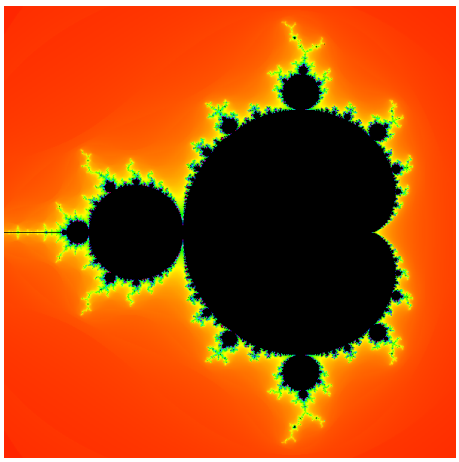
In dieser Aufgabe schreiben Sie ein Programm, dass die Mandelbrotmenge darstellt.
Die Mandelbrotmenge ist definiert als die Teilmenge der komplexen Zahlen c, für die die Folge $z_0 = 0, z_{n+1} = z_n^2 + c$ beschränkt ist. Das heisst, die Mandelbrotmenge besteht aus den Zahlen, für die es eine Konstante 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 $z_0 = 0, z_{n+1} = z_n^2 + c$ berechnet, solange $|z_n| < 2$ ist. Dann soll ein MandelbrotResult Objekt zurück gegeben werden, welches die Resultate der Berechnung enthält.
Testen Sie Ihr Programm mit den mitgelieferten Tests.

Aufgabe 9.4 (Visualisierung der Mandelbrotmenge)

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 9.5 (Bonus: Kameraflug)

In dieser Aufgabe müssen Sie nicht selber etwas implementieren. Diese Aufgabe ist zum geniessen der Schönheit der Mandelbrotmenge. Kopieren Sie dazu den folgenden Code in die Main Methode. Öffnen Sie das Bild in der IDE um zu sehen, wie sich das Bild verändert während das Programm läuft.

```
Mandelbrot mb = new Mandelbrot();
for ( double d = 0.004; d > 1e-10; d *= 0.9){
    mb.createMandelbrotVisualization(
        d,
        -0.604894,
        -0.614897,
        100
    );
    mb.save("mandelbrot.png");
    try {
        Thread.sleep(100);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
```

Abgabe Erstellen Sie eine Zip-Datei der gesamten Übungsumgebung (also des Verzeichnisses `uebung009`) und laden Sie dieses auf Adam hoch.