Praktikum Programmieren
Studiengang Angewandte Informatik
Prof. Dr. Bernd Kahlbrandt
Dipl.-Ing. Gerhard Oelker
Dipl.-Wirtschaftsinformatiker (FH) Sven Berding
Hochschule für Angewandte Wissenschaften Hamburg
Departement für Informatik
19. März 2018

Aufgabenblatt 01: Planung für das Semester und erste Aufgaben

1 Ideen für das Praktikum dieses Semesters

Bearbeiten Sie die Praktikumsaufgaben bitte in Zweierteams!

Eine Idee für ein kleines Projekt im Praktikum einer Java-Veranstaltung ist so naheliegend, gerade wenn Ruby bereits bekannt ist, dass ich nicht darauf verzichten mochte. Auch Oracle verwendet elementare Teile davon in den Demo-Anwendungen für JavaFX. Eine Folge von Aufgaben plane ich schon, weitere können dazu kommen:

1. In *Ruby* gibt es eine Klasse *Complex*, in *Java* nicht. In *Ruby* sind Rechenoperationen (+, -, ∗, /) als Methoden implementiert, in *Java* sind dies Operatoren, die im Unterschied zu *C++* nicht überladen werden können.

   Es könnte also eine gute Übung sein, die Ruby-Klasse in eine Java-Klasse zu „übersetzten“. Ziel dieser Folge von Aufgaben ist es, am Ende des Semesters eine getestete, stabile, für Produktionsanwendungen einsetzbare Klasse *Complex* zu haben.

2. Vielleicht kennen einige von Ihnen die sog. Mandelbrotmenge. Dies ist ein Beispiel eines Fraktals. Um diese zu berechnen ist eine Klasse *Complex* nützlich. Die Fraktale sollen graphisch dargestellt werden. Dazu werde ich versuchen Ihnen JavaFX nahe zubringen. Je nachdem, wie weit wir im Stoff kommen, kann dabei auch die Programmierung paralleler Anwendungen in Java geübt werden. In den Demo-Anwendungen für JavaFX finden Sie dies unter MandelbrotSet. Das nur für den Fall, dass Sie schon einmal spielen wollen.

   Am Ende dieses Aufgabenblatts finden Sie eine Auswahl von Büchern über Fraktale. Ich bitte um Info, welche davon Ihnen in der elektronischen Bibliothek zur Verfügung stehen.

3. Nach den ersten Grundlagen der Fraktalanwendung, werde ich versuchen die Aufgaben auf mehrere Schultern zu verteilen: Jedes Team soll andere Fraktale ergänzen.

4. Bevor wir mit dem GUI-Teil beginnen können, benötigen Sie aber einige weitere Grundlagen:

- Generics
- Annotationen
- Lambda-Ausdrücke
- Exceptions (werden Sie schnell lernen, ist ganz ähnlich wie in Ruby).

Zu diesen Themen wird es einzelne Aufgaben geben, um isoliert die Techniken zu üben.

2 Klasse Complex

In diesem Aufgabenstrang sollen Sie eine produktiv einsetzbare Klasse *Complex* schreiben und zugehörige Methoden in einer Utility-Klasse. Die ersten Versionen werden noch nicht alle Funktionalität bieten können, aber allerspätestens am Ende des Semesters sollten beide Klassen ausgetestet zur Verfügung stehen. Ich strebe an, dass wir am Ende des Semesters eine Entscheidung über die beste Implementierung treffen.

Dokumentieren Sie bitte alle Elemente Ihrer Klassen mit Javadoc!
Im Einzelnen:

1. Schreiben Sie bitte zwei Versionen einer Klasse *Complex*, deren Objekte komplexe Zahlen repräsentieren, eine mutable und eine immutable! Sie können zwei Klassen gleichen Namens in zwei Paketen schreiben, aber das ist nicht die einzige, vielleicht auch nicht die beste Möglichkeit. Sie können und sollen von allen sinnvollen Möglichkeiten der Wiederverwendung Gebrauch machen (Interface, abstrakte Klasse, ...).

   Die anderen Klasse soll folgenden Eigenschaften haben:

   1.1. Die üblichen Rechenmethoden für komplexe Zahlen. Sie können sich dabei an der Klasse *Complex* aus Ruby orientieren, aber bleiben Sie bitte kritisch.

   1.2. Es sollen komplexe Zahlen mit reellen cartesischen Koordinaten und mit reellen Polarkoordinaten erzeugt werden können!

   1.3. Für eine komplexe Zahl sollen sowohl die kartesischen als auch die Polarkoordinaten über Methoden abgefragt werden können.

   1.4. Denken Sie bitte auch an Methoden, wie den absoluten Betrag einer komplexen Zahl, die konjugiert komplexe Zahl $\bar{z}$, etc.

   1.5. Überschreiben Sie bitte *equals*, *hashCode* und *toString* auf sinnvolle Weise!

2. Schreiben Sie bitte ein Utility-Klasse *MathUtils* in der Sie einige wichtige mathematische Funktionen für komplexe Zahlen definieren. Ich nenne als Beispiele Exponentialfunktion, Sinus, Cosinus.

   Klassenmethoden werden in *Java* durch das Schlüsselwort *static* gekennzeichnet.

3. Demonstrieren Sie an Hand geeigneter JUnit-Testfälle, dass Ihre komplexen Zahlen und die Methoden aus MathUtils funktionieren.

Abgabetermin:

Donnerstag, 12.04.2018.

Literatur

[1] Hans Lauwerier. *Fraktale verstehen und selbst programmieren. Band 1: Einführung.* Wittich Fachbuchverlag, Hückelhoven, 2., verbesserte Auflage, 1992, 197 Seiten. ISBN 3-88984-060-4.

[2] Hans Lauwerier. *Fraktale verstehen und selbst programmieren. Band 1: Vertiefung.* Wittich Fachbuchverlag, Hückelhoven, 2., verbesserte Auflage, 1992, 270 Seiten. ISBN 3-88984-061-2.

[3] Benoît B. Mandelbrot. *Die fraktale Geometrie der Natur.* Birkhäuser, Basel, Boston, Berlin, Einmalige Sonderausgabe Auflage, 1991, 491 Seiten. ISBN 3-7643-2646-8.

[4] Hans Meinhardt. *The Algorithmic Beauty of Sea Shells.* Springer-Verlag, Berlin, Heidelberg, New York, NY, 3. Auflage, 2003, xi+236 Seiten. ISBN 3-540-44010-0.

[5] Heinz-Otto Peitgen, Hartmut Jürgens und Dietmar Saupe. *Bausteine des Chaos – Fraktale.* Springer, Klett-Cotta, Berlin, Heidelberg, New York, Stuttgart, 1992.

[6] Heinz-Otto Peitgen, Hartmut Jürgens und Dietmar Saupe. *Fractals for the Classroom – Part Two: Complex Systems and the Mandelbrot Set*, Band 2. Springer-Verlag, Berlin, Heidelberg, New York, NY, 1992.

[7] Heinz-Otto Peitgen und Peter H. Richter. *The Beauty of Fractals.* Springer-Verlag, Berlin, Heidelberg, New York, NY, 1986.