Praktikum Programmieren

Aufgabenblatt 1 - Entwurf

B-AI2 PMP SS 2018

Adrian Helberg, Gruppe 2

Prüfer: Prof. Dr. Bernd Kahlbrandt

11. April 2018

Interface Complex

Klasse ImmutableComplex

Klasse MutableComplex

Klasse MathUtils

JUnit Tests

Instanz-Methoden

Interface Complex

Das Interface *Complex* soll als Basis für die Erstellung einer *"immutable"* und einer *"mutable"* Repräsentation von komplexen Zahlen dienen.
Da Interfaces in Java 8 Default-Implementationen bietet, wird in diesem Projekt nicht auf eine abstrakte Klasse Complex zurückgegriffen, da so viele Zeilen Code gespart werden können.

Interface Complex

Das Interface wird wie folgt zusammengesetzt:

- Diverse abstrakte Funktionssignaturen, die von erbenden Klassen überschrieben werden müssen
- Default-Implementationen für *"immutable"* Objekte, die nur von *mutable* Objekten überschrieben werden müssen, da diese eine andere Implementation brauchen
- Dokumentation in Form von **Javadoc**

Klasse ImmutableComplex

Die Klasse *ImmutableComplex* erzeugt nicht veränderbare (**final**) Objekte als komplexe Zahl. Die Instanzmethoden geben immer ein neuen Objekt zurück, wenn bestimmte Operationen auf den Objekten ausgeführt werden.
Die Klasse implementiert das Interface *Complex* und besteht aus

- Diversen Konstruktoren
- Überschriebenen Instanzmethoden
- Allen übrigen Instanzmethoden, die eine Default-Implementation im zu erbenden Interface besitzen

Klasse MutableComplex

Die Klasse *MutableComplex* erzeugt veränderbare Objekte als komplexe Zahl. Die Instanzmethoden verändern die aktuelle Instanz (**this**) und geben, wenn bestimmte Operationen auf den Objekten ausgeführt werden, diese anschließend zurück.
Die Klasse implementiert das Interface *Complex* und besteht aus

- Diversen Konstruktoren
- Überschriebenen Instanzmethoden
- Allen übrigen Instanzmethoden, die nicht überschrieben wurden und eine Default-Implementation im zu erbenden Interface besitzen

Klasse MathUtils

Die Klasse *MathUtils* stellt diverse statische Funktionen für komplexe Zahlen zur Verfügung. Unter anderem aber auch eine *round*-Funktion, um auf eine bestimmte Anzahl Nachkommastellen zu runden. Da es sich bei den komplexen Zahlen um veränderbare und nicht veränderbare Objekte handelt, wurden einige Funktionen mit dem *Reflection-Tool* **instance of** implementiert.

JUnit Tests

Alle implementierten Konstruktoren, Funktionen und Methoden werden mit *JUnit* **assertions** in den Test-Klassen

- *ImmutableComplexTest*
- *MutableComplexTest*
- *MathUtilsTest*

getestet.

Instanz-Methoden

Folgende Instanz-Methoden werden implementiert [1]

- *getRe*
- *getIm*
- *getR*
- *getTheta*
- *multiply*
- *add*
- *subtract*
- *reciprocal*
- *negate*

[1] siehe https://ruby-doc.org/core-2.3.0/Complex.html

Instanz-Methoden

- *divide*
- *equals*
- *abs*
- *abs2*
- *conjugate*
- *hash*
- *formatCartesian*
- *formatTrigonometric*
- *formatPolar*