University of Zurich
Department of Informatics

s.e.a.l.
software evolution & architecture lab

Programmierung für Mathematik HS10

Übung 2

1 Aufgabe: Calculator Fortsetzung

1.1 Lernziele

1. Probleme mit Integer-Rechnung kennenlernen und Lösungen dafür finden.
2. Komplexe Zahlen in Java-Code realisieren können.

1.2 Aufgabenstellung

a) grosse Zahlen addieren

Testen Sie die Methode `add` von der Klasse `Calculator` vom Übungsblatt 1 mit den Zahlen 2000000000 und 1000000000. Was fällt Ihnen auf und wie erklären Sie sich das? Was wäre eine einfache Lösungsmöglichkeit für dieses Problem?

b) Division von Integervariabeln

Erweitern Sie Ihre `Calculator`-Klasse um die Methode `divide`, welche den Quotient zweier Zahlen vom Typ `int` berechnen soll. (Das Resultat soll als Dezimalzahl angezeigt werden, man darf voraussetzen, dass der Benutzer als zweite Zahl nicht 0 eingibt.) Testen Sie Ihre Methode mit den Zahlen 5 und 2.

c) Rechnen mit komplexen Zahlen

Gegeben ist die Klasse `ComplexNumber`, welche eine komplexe Zahl $a + ib$ darstellen soll.

```
1  public class ComplexNumber{
2
3     private double a;
4     private double b;
5
6     public double getA() {
7        return a;
8     }
9     public void setA(double newA) {
10       a = newA;
11    }
12    public double getB() {
13       return b;
14    }
15    public void setB(double newB) {
16       b = newB;
17    }
18    public String toString() {
19       String result = a + " + i*" + b;
20       return result;
21    }
22 }
```

Listing 1: `ComplexNumber` Klasse

Speichern Sie diese Klasse unter *ComplexNumber.java* ab. Verändern Sie die Klasse `Calculator` so, dass die Methoden `add`, `subtract`, `multiply` und `divide` für komplexe Zahlen verwendet werden können. (das heisst die Objekte, die den Methoden übergeben werden, sollen nun nicht mehr vom Typ `int`, sondern vom Typ `ComplexNumber` sein.)
Bemerkung zum Rechnen mit komplexen Zahlen: Eine komplexe Zahl hat die Form $a + i \cdot b$, wobei $a, b \in \mathbb{R}$ und $i^2 := -1$. Damit kann nun 'wie bekannt' gerechnet werden, also so, als wäre i eine beliebige Variable. Beispiel Division (es sei $(c, d) \neq (0, 0)$ und $a, b, c, d \in \mathbb{R}$):

$$\begin{aligned}
\frac{a+i\cdot b}{c+i\cdot d} &= \frac{(a+i\cdot b)(c-i\cdot d)}{(c+i\cdot d)(c-i\cdot d)} = \frac{ac-i\cdot ad+i\cdot bc-i^2\cdot bd}{c^2-(i\cdot d)^2} \\
&= \frac{ac-i\cdot ad+i\cdot bc-(-1)\cdot bd}{c^2-i^2\cdot d^2} = \frac{(ac+bd)+i\cdot(bc-ad)}{c^2-(-1)\cdot d^2} \\
&= \frac{ac+bd}{c^2+d^2}+i\cdot\frac{bc-ad}{c^2+d^2} \in \mathbb{C}
\end{aligned}$$

2 Aufgabe: Vektorrechnen im $\mathbb{R}^3$

2.1 Lernziele

1. Verschiedene Arten von Lösungsausgaben von Methoden implementieren können.
2. Mathematische Formeln in Java implementieren können.

2.2 Aufgabenstellung

Die Klasse `Vector` soll einen Vektor $\begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$ darstellen und ist wie folgt gegeben.

```
1  public class Vector {
2
3     private double x,y,z;
4
5     public double getX() {
6        return x;
7     }
8     public void setX(double newX) {
9        x = newX;
10    }
11    public double getY() {
12       return y;
13    }
14    public void setY(double newY) {
15       y = newY;
16    }
17    public double getZ() {
18       return z;
19    }
20    public void setZ(double newZ) {
21       z = newZ;
22    }
23
24    public void add1(Vector vec){
25
26    }
27
28    public void add2(Vector vec){
29
30    }
31
32    public Vector vectorProduct(Vector vec){
33
34    }
35 }
```

Listing 2: `Vector` Klasse

a) Addition von Vektoren

Ergänzen Sie die Methoden `add1` und `add2` dahingehend, dass die Summe von dem übergebenen Vektor und dem Vektor, auf welchem die Methode aufgerufen wurde, berechnet werden.

1. `add1` soll das Resultat in dem Vektor speichern, der bei Methodenaufruf übergeben werden muss (also in `vec`).
2. `add2` soll das Resultat in dem Vektor speichern, auf welchem die Methode `add2` aufgerufen wird.

b) Vektorprodukt

Ergänzen Sie die Methode `vectorProduct`. Die Methode soll ein neues Objekt vom Typ `Vector` erstellen, in dem das Vektorprodukt (von dem methodenaufrufenden Vektor und `vec`) gespeichert wird, und von der Methode als `return`-Wert zurückgegeben wird.
Formel für das Vektorprodukt:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_2y_3 - x_3y_2 \\ x_3y_1 - x_1y_3 \\ x_1y_2 - x_2y_1 \end{pmatrix}$$

3 Aufgabe: Code Snippets

3.1 Lernziele

1. Java API kennenlernen.
2. Code verstehen können.

3.2 Aufgabenstellung

Überlegen Sie sich, was der Output bei folgenden Codestücken wäre, wenn sie so in einer `main`-Methode stehen würden.
Tipp: benutzen Sie die Java API um die vorimplementierten Java-Methoden zu verstehen, in dem folgenden Code werden Methoden von den Klassen `Math`, `String` und `Character` verwendet.

```
1 int a = 2000;
2 int b = a;
3 a = a + 2;
4 System.out.println(a);
5 System.out.println(b);
```

Listing 3:

```
1 String s1 = "animal";
2 String s2 = "dog";
3 System.out.println(s2+(Character.toString(s1.charAt(2))).concat("s")+s1.
      substring(0,2)+s1);
```

Listing 4:

```
1 System.out.println(Math.PI);
```

Listing 5:

```
1 System.out.println(Math.round(20/7));
2 System.out.println(Math.round(20.0/7));
3 System.out.println(Math.ceil(20.0/7));
4 System.out.println(Math.floor(20.0/7));
```

Listing 6:

```
1 System.out.println(Math.log(1));
2 System.out.println(Math.exp(0));
3 System.out.println(Math.cos(Math.PI));
```

Listing 7: