Übungsblatt 4:

Aufgabe 1: Überladen von Methoden

Gegeben sei die folgende Klasse, die mehrere überladene Methoden `tuWas` mit unterschiedlichen Signaturen enthält:

```
public class Signatur {
   public static void tuWas(int x, int y, int z) {
      System.out.println("iii");
   }
   public static void tuWas(int x, int y, double z) {
      System.out.println("iid");
   }
   public static void tuWas(int x, double y, double z) {
      System.out.println("idd");
   }
   public static void tuWas(double x, int y, double z) {
      System.out.println("did");
   }
   public static void main (String[] args) {
      int a = 123;
      double b = 4.56;
      tuWas(a,a,a); // Aufruf 1
      tuWas(a,a,b); // Aufruf 2
      tuWas(a,b,a); // Aufruf 3
      tuWas(a,b,b); // Aufruf 4
      tuWas(b,b,b); // Aufruf 5
      tuWas(b,a,b); // Aufruf 6
      tuWas(b,a,a); // Aufruf 7
   }
}
```

Geben Sie für alle 7 Methoden-Aufrufe in der `main`-Methode an, ob diese zulässig oder unzulässig sind. Geben Sie bei zulässigen Aufrufen an, was auf dem Bildschirm ausgegeben wird.

Schauen Sie sich nun die Klasse `Signatur2` an. Was passiert hier?

```
public class Signatur2 {
   public static void tuWas(int x, int y, double z) {
      System.out.println("iid");
   }
   public static void tuWas(int x, double y, int z) {
      System.out.println("idi");
   }
   public static void main (String[] args) {
      int a = 123;
      double b = 4.56;
      tuWas(a,a,a);
   }
}
```

Aufgabe 2: Referenzvariablen

In folgendem Programm wird ein Feld von Referenzvariablen angelegt. Dazu werden drei Instanzen der Klasse ObjRef generiert. Finden Sie heraus, welche Referenzvariablen am Ende auf (welche) Objekte verweisen.

```
public class ObjRef {
    int id =0;
    public static void main(String[] args){
        int x =0;
        ObjRef[] hq = new ObjRef[5];
        while (x<3){
            hq[x] = new ObjRef();
            hq[x].id = ++x;
        }
        hq[3] = hq[1];
        hq[4] = hq[1];
        hq[3] = null;
        hq[4] = hq[0];
        hq[0] = hq[3];
        hq[3] = hq[2];
        hq[2] = hq[0];
```

```
    }
}
```

Aufgabe 3: Call by Value / Call by Reference

Geben Sie an, was bei der Ausführung der Methode `main` der Klasse `AchSo` ausgegeben wird.

```
public class AchSo {
   public static void ach (int i, int[] j) {
      i= i * 2;
      j[0] = i - 2;
      System.out.println("ach: " + i + " " + j[0]);
   }
   public static void so (int[] i, int j) {
      i[0] = i[0] - j;
      j= i[0] * 2;
      System.out.println("so: " + i[0] + " " + j);
   }
   public static void main (String[] args) {
      int[] m = {5};
      int[] n = {7};
      System.out.println("main: " + m[0] + " " + n[0]);
      ach(m[0],n);
      System.out.println("main: " + m[0] + " " + n[0]);
      so(m,n[0]);
      System.out.println("main: " + m[0] + " " + n[0]);
      }
   }
```

Aufgabe 4: Referenzen

Die Klasse `Form` definiert eine geometrische Struktur, die sich aus Strecken zusammensetzt. Jede Strecke besteht wiederum aus einem Anfangs- und einem Endpunkt. Dabei referenzieren Objekte der Klassen `Strecke` und `Form` jeweils wieder ihre Bestandteile.
Analysieren Sie folgendes Programm und skizzieren Sie alle Referenzvariablen und die von ihnen jeweils referenzierten Objekte.
Zeichnen Sie dann die durch das `form`-Objekt repräsentierte Figur, wie diese jeweils an den markierten Stellen im Code aussehen sollte.

```
class Punkt {
    public double x;
    public double y;
}

class Strecke {
    public Punkt from;
    public Punkt to;
}
public class Form {
    public Strecke[] streckenzug;

    public static void main(String[] args){
        Punkt p1 = new Punkt();
        p1.x = 0;
        p1.y = 0;
        Punkt p2 = new Punkt();
        p2.x = 5;
        p2.y = 0;
        Punkt p3 = new Punkt();
        p3.x = 5;
        p3.y = 5;
        Punkt p4 = new Punkt();
        p4.x = 0;
        p4.y = 5;
```

```
        Strecke s1 = new Strecke();
        s1.from = p1;
        s1.to = p2;
        Strecke s2 = new Strecke();
        s2.from = p2;
        s2.to = p3;
        Strecke s3 = new Strecke();
        s3.from = p3;
        s3.to = p4;
        Strecke s4 = new Strecke();
        s4.from = p4;
        s4.to = p1;

        Form form = new Form();
        form.streckenzug = new Strecke[] {s1,s2,s3,s4};
        // *** hier ***
        form.streckenzug[1].to.y = 3;
        form.streckenzug[3].from.y = 3;
        // *** und hier ***
    }
}
```

Aufgabe 5: Statische Methoden

Schreiben Sie eine Klasse `Rechnen`, die mehrere statische Methoden besitzt, welche folgende mathematische Operatoren für ganze Zahlen implementieren sollen:

abs Diese Methode soll den Absolutbetrag einer als Parameter übergebenen Zahl berechnen und zurückgeben.

neg Diese Methode soll den Wert der als Parameter übergebenen Zahl mit (-1) multiplizieren und zurückgeben.

sqr Diese Methode soll den Wert der als Parameter übergebenen Zahl quadrieren und zurückgeben

pow Diese Methode soll die als ersten Parameter übergebene Zahl zu der als zweiter Parameter bergebenen Zahl potenzieren (also a^b rechnen).

harmSum Diese Methode soll die harmonische Reihe bis zum `n`. Summanden berechnen (d.h. $\sum_{i=1}^{n} \frac{1}{i}$)

fak Diese Methode soll für eine als Parameter übergebenen Zahl n den Wert $n! = \Pi_{i=1}^{n} i$ berechnen. Finden Sie heraus, wie groß n werden darf, wenn das Ergebnis vom Typ `int` bzw. `long` ist.

max Diese Methode soll das Maximum der Werte in einem als Parameter übergebenen Feld bestimmen und zurückgeben.

Achten Sie darauf, dass die Rückgabewerte (soweit möglich) ausreichend groß sind und geben Sie bei ungültigen Argumenten eine Fehlermeldung aus.Der Rückgabewert im Fehlerfall sollte dann eine passende Zahl sein, der für einen Fehler stehen kann.
Testen Sie nun Ihre Klassenmethoden mit Hilfe der Klasse `TestRechnen`, die nur eine main-Methode enthält, in der verschiedene Variablen deklariert, eingelesen und mit denen dann die Methoden der Klasse `Rechnen` aufgerufen werden. Eine Ausgabe könnte dann z.B. wie folgt aussehen:

```
-->Zahl 1: -2
-->Zahl 2: 4
Absolutbetrag von -2 = 2
Negation von  -2 = 2
Quadrat von -2 = 4
-2 hoch 4 = 16
-->n: 5
1 + 1/2 + ... + 1/5 = 2.283333333333333
1 * 2 * ... * 5 = 120
-->Feldlaenge: 6
-->feld[0] = -1
-->feld[1] = 3
-->feld[2] = 7
-->feld[3] = 9
-->feld[4] = 0
-->feld[5] = -5
Maximum ist: 9
```

Aufgabe 6: Instanzmethoden

Schreiben Sie eine Klasse für komplexe Zahlen. Komplexe Zahlen treten z.B. auf, wenn man die Quadratwurzel aus einer negativen Zahl zieht. Komplexe Zahlen verfügen über zwei Komponenten, die beide als double Werte repräsentiert werden sollen. Der eine Wert ist der sogenannte Realteil der Komplexen Zahl, der andere der sogenannte Imaginärteil. Diese beiden Komponenten bilden die Attribute der Klasse.

Die Darstellung einer solchen Komplexen Zahl sieht folgendermassen aus:

$$a + bi$$

Dabei bildet a den Realteil der Zahl und b den Imaginärteil. Das Symbol i steht für das Ergebnis der Quadratwurzel aus -1 ($\sqrt{-1}$). Detailliertere Informationen zu Komplexen Zahlen finden Sie in Mathematikbüchern, oder z.B. auf Wikipedia.

- Überlegen Sie sich zunächst, welche Konstruktoren sinnvoll wären und implementieren Sie diese.
- Implementieren Sie eine Methode, die eine String-Repräsentation einer Komplexen Zahl in der Form $a + bi$ generiert und als Ergebnis zurückgibt.
- Fügen Sie nun noch einige arithmetische Operatoren für Komplexe Zahlen hinzu: Die Rechenregeln für Komplexe Zahlen sind dabei wie folgt:
  Addition: $(a + bi) + (c + di) = (a + c) + (b + d)i$

  Subtraktion: $(a + bi) - (c + di) = (a - c) + (b - d)i$

  Multiplikation: $(a+bi)*(c+di) = (a*c)+(c*b)i+(a*d)i+(b*d)i^2 = (a*c-b*d)+(c*b+a*d)i$

  Division: $\frac{a+bi}{c+di} = \frac{(a+bi)*(c+di)}{(c+di)*(c+di)} = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}i$ Die Symbole a und c stehen hierbei immer für die Realteile der beiden Komplexen Zahlen und b und d für die Imaginärteile. i repräsentiert $\sqrt{-1}$. Schreiben Sie für jede Operation eine entsprechende Methode, die stets die in der jeweiligen Instanz repräsentierte Komplexe Zahl als ersten Operanden

verwendet und den zweiten Operanden als Parameter übergeben bekommt. Das Resultat soll jeweils in Form einer neuen Komplexen Zahl zurückgegeben werden.

- Erweitern Sie nun die Klasse um eine `main`-Methode, in der Sie zwei Komplexe Zahlen erzeugen, die oben genannten Operationen darauf ausführen und die Ergebnisse ausgeben. Die Ausgabe bei Aufruf des Programs könnte dann beispielsweise wie folgt aussehen:

```
-->Realteil 1. Zahl: 3
-->Imaginaerteil 1. Zahl: -1
-->Realteil 2. Zahl: 4
-->Imaginaerteil 2. Zahl: -3
Erste Zahl: 3.0 + -1.0 * i
Zweite Zahl 4.0 + -3.0 * i
z1 + z2 = 7.0 + -4.0 * i
z1 - z2 = -1.0 + 2.0 * i
z1 * z2 = 9.0 + -13.0 * i
z1 / z2 = 0.6 + 0.2 * i
```