Prof. Dr. M. Müller C. Aschermann, J. Hensel, J. Protze, P. Reble

Allgemeine Hinweise:

- Die **Hausaufgaben** sollen in Gruppen von je **3 Studierenden** aus der **gleichen Kleingruppenübung (Tutorium)** bearbeitet werden. **Namen und Matrikelnummern** der Studierenden sind auf jedes Blatt der Abgabe zu schreiben. **Heften bzw. tackern Sie die Blätter!**
- Die **Nummer der Übungsgruppe** muss **links oben** auf das **erste Blatt** der Abgabe geschrieben werden. Notieren Sie die Gruppennummer gut sichtbar, damit wir besser sortieren können.
- Die Lösungen müssen **bis Montag, den 07.12.2014 um 15:00 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55). Alternativ können Sie die Lösungen auch vor der Abgabefrist direkt bei Ihrer Tutorin/Ihrem Tutor abgeben.
- In einigen Aufgaben müssen Sie in Java programmieren und `.java`-Dateien anlegen. **Drucken** Sie diese aus **und** schicken Sie sie per **E-Mail** vor Montag, dem 07.12.2014 um 15:00 Uhr an Ihre Tutorin/Ihren Tutor.
  Stellen Sie sicher, dass Ihr Programm von `javac` **akzeptiert** wird, ansonsten werden keine Punkte vergeben.

Tutoraufgabe 1 (Programmanalyse):

Lösen Sie die folgende Aufgabe ohne Einsatz eines Computers. Bedenken Sie, dass Sie in einer Prüfungssituation ebenfalls keinen Computer zur Verfügung haben.
Betrachten Sie das folgende kurze Programm:

```
public class A {

    private Integer i;

    private double d;

    public A() {
        this.i = 1;
        this.d = 4;
    }

    public A(Integer x, double y) {
        this.i = x;
        this.d = y;
    }

    public A(int x, double y) {
        this.i = 3;
        this.d = x + y;
    }

    public int f(Integer x) {
        return this.i + x;
    }

    public int f(double i) {
        this.d = i;
        return this.i;
```

```
    }

    public static void main(String[] args) {
        A a1 = new A();
        System.out.println(a1.f(5));
        System.out.println(a1.d);
        System.out.println(a1.f(new Long(2)));
        A a2 = new A(1,1);
        System.out.println(a2.i);
        System.out.println(a2.d);
    }

}
```

Geben Sie die Ausgabe dieses Programms an. Begründen Sie Ihre Antwort.

Aufgabe 2 (Programmanalyse): (14,5 Punkte)

Lösen Sie die folgende Aufgabe ohne Einsatz eines Computers. Bedenken Sie, dass Sie in einer Prüfungssituation ebenfalls keinen Computer zur Verfügung haben.
Betrachten Sie das folgende kurze Programm:

```
public class B {
        private static int cnt = 0;
        private double d1;
        private Float f;
        private Integer a;
        private int j;
        public B(int a, int b, int c, int d) {
                this.a = a;
                this.j = b;
                this.d1 = d;
                this.f = (float)c;
                cnt++;
        }
        public B(Integer a, int b, double c, Float d) {
                this.a = a;
                this.j = b;
                this.d1 = c;
                this.f = d;
                cnt++;
        }
        public int f(Float x, int y) { return 101; }
        public int f(double x, int y) { return 102; }
        public Integer f(double x, Long y) { return 103; }
        public double g(long x) { return 7.0; }
        public Float g(Integer x) { return 8f; }
        public static void main(String[] args) {
                B b1 = new B(1,2,3,4);
                System.out.println(b1.d1);
                System.out.println(b1.f(7.5f,8));
                System.out.println(b1.f(5,6));
                System.out.println(b1.f(10.2f,17L));
                B b2 = new B(b1.a, 5, 6, 9);
                System.out.println(b2.f);
                System.out.println(b2.f(b1.f,b1.j));
                B b3 = new B(++b2.a, 14, 15, 16f);
                System.out.println(b3.d1);
                System.out.println(b3.g(new Long(18+1)));
                System.out.println(b3.g(b1.a));
                System.out.println(b3.f(b2.g(19), 21));
                System.out.println(b1.j++);
                System.out.println(cnt);
        }
}
```

Geben Sie die Ausgabe dieses Programms an. Begründen Sie Ihre Antwort. Geben Sie zusätzlich die Werte aller Attribute am Programmende an (Bei Objekten von Wrapper-Klassen genügt der Wert des eingehüllten Primitiven Datentyps).

Tutoraufgabe 3 (Einfache Rekursion):

Betrachten Sie folgende Methode:

```
public static int gauss(int n) {
    int res = 0;
    int i = 1;
    while (i <= n) {
        res += i;
        i++;
    }
    return res;
}
```

Schreiben Sie eine statische Methode `gaussRecursive`, welche eine `int`-Zahl erhält und eine `int`-Zahl zurückliefert, sodass für jede `int`-Zahl `x` die Aufrufe `gauss(x)` und `gaussRecursive(x)` das gleiche Ergebnis liefern. Sie dürfen in dieser Aufgabe keine Schleifen verwenden. Die Verwendung von Rekursion ist hingegen erlaubt.

Aufgabe 4 (Rekursion): (4 Punkte)

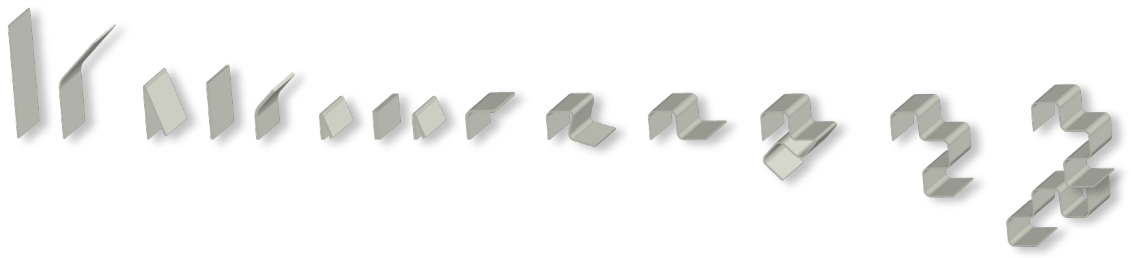
Betrachten Sie folgende Methode:

```
public static int arraySum(int[] a) {
    int res = 0;
    for (int i = 0; i < a.length; i++) {
        res += a[i];
    }
    return res;
}
```

Schreiben Sie eine statische Methode `arraySumRecursive`, welche ein `int`-Array erhält und eine `int`-Zahl zurückliefert, sodass für jedes `int`-Array `a` die Aufrufe `arraySum(a)` und `arraySumRecursive(a)` das gleiche Ergebnis liefern. Sie dürfen in dieser Aufgabe keine Schleifen verwenden. Die Verwendung von Rekursion ist hingegen erlaubt (inklusive der Definition von Hilfsmethoden).

Tutoraufgabe 5 (Drachenkurve):

Ziel dieser Aufgabe ist es, ein Programm zur Visualisierung der Drachenkurve zu schreiben. Die Drachenkurve erhält man, indem man einen Papierstreifen wie folgt faltet: Man falte die beiden Enden des Papierstreifens so aufeinander, dass sich die Länge des Streifens halbiert. Den so verkürzten Streifen faltet man wieder auf die selbe Art (und—wichtig—in die selbe Richtung). Nach ein paar Wiederholungen faltet man den Streifen wieder auseinander und richtet ihn dann so aus, dass jeder Knick genau rechtwinklig verläuft. Hat man den Streifen anfänglich n-mal halbiert, erhält man durch diese Anleitung eine Drachenkurve der n-ten Ordnung.



[1]

Diese Drachenkurve n-ter Ordnung lässt sich auch durch ihre Folge von **R**echts- bzw. **L**inksknicken beschreiben:

[1] Bild lizenziert unter CC BY-SA 3.0, Quelle:
`http://de.wikipedia.org/w/index.php?title=Datei:Dragon_curve_paper_strip.png`

Ordnung	Folge
1	R
2	R **R** *L*
3	R R L **R** R *L* L
4	R R L R R L L **R** R R L *L* R L L
...	...

Hierbei ergibt sich folgende Konstruktionsvorschrift: Eine Drachenkurve der 1-ten Ordnung hat nur einen **R**echtsknick. Um eine Drachenkurve der Ordnung $(n+1)$ zu erhalten, führt man erst die Folge von Knicken einer Kurve der Ordnung n durch, dann einen **R**echtsknick und dann wieder die Knicke einer Folge der Ordnung n, wobei man in deren Mitte anstelle eines **R**echtsknicks einen **L**inksknick macht.
Für Ihre Implementierung benötigen Sie die Datei `Staffelei.java` von der Homepage. Verwenden Sie das Programm Javadoc, um die Schnittstellendokumentation dieser Klasse zu erzeugen.
Ergänzen Sie die Datei `Drachenkurve.java` um zwei statische Methoden `kurveL` und `kurveR`, die jeweils eine Drachenkurve beliebiger Ordnung mit einem Links- bzw. Rechtsknick in der Mitte darstellen. Diese Methoden bekommen als Argumente jeweils ein Objekt der Klasse `Staffelei` sowie eine Ordnung als `int`-Wert. Mit Hilfe der Methoden des `Staffelei`-Objektes kann die Zeichnung gefertigt werden.

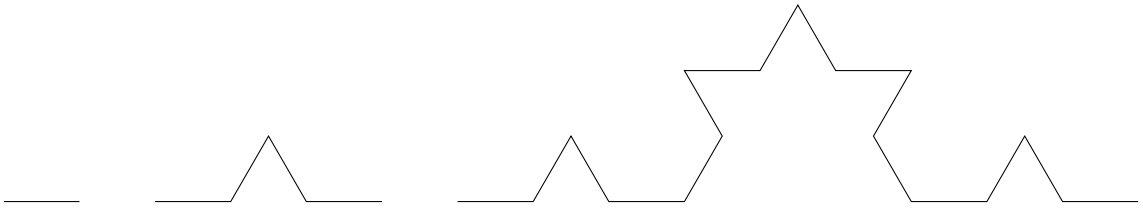
Hinweise:

- Verwenden Sie die Methode `drawForward` der Klasse `Staffelei` um einen Strich zu malen. Als Länge eignen sich 10 Pixel.
- Rotationen können Sie mit der Methode `rotate` der Klasse `Staffelei` erreichen, die eine Gradzahl als Parameter bekommt.
- Implementieren Sie die Methoden `kurveL` und `kurveR` rekursiv.

Aufgabe 6 (Kochkurve): (5 Punkte)

Ziel dieser Aufgabe ist es, ein Programm zur Visualisierung der Kochkurve zu programmieren, die sich ähnlich beschreiben lässt wie die Drachenkurve der vorhergehenden Aufgabe:

- Eine Kochkurve der Ordnung 0 ist eine gerade Linie einer bestimmten Länge.
- Eine Kochkurve der Ordnung $n+1$ sind vier Kochkurven der Ordnung n, die nacheinander gezeichnet werden. Nach der ersten und dritten Kurve wird ein Knick um 60° nach links gemacht und nach der zweiten Kurve ein Knick um 120° nach rechts.



Kochkurven der Ordnung 0, 1 und 2.

Für Ihre Implementierung benötigen Sie die Datei `Staffelei.java` von der Homepage. Verwenden Sie das Programm Javadoc, um die Schnittstellendokumentation dieser Klasse zu erzeugen.
Ergänzen Sie die Datei `Kochkurve.java` um eine statische Methode `kochkurve`. Diese Methode soll eine Kochkurve beliebiger Ordnung darstellen können. Als Eingabeparameter bekommt die Methode ein Objekt der Klasse `Staffelei` sowie die Ordnung der darzustellenden Kurve als `int`-Wert.
Die (fertige) `main`-Methode in der Klasse `Kochkurve` erstellt erst ein Objekt der Klasse `Staffelei` und verwendet dann die Funktion `kochkurve`, um drei Kochkurven der Ordnung 3 zu malen, die jeweils durch eine Rechtsrotation von 120° verbunden sind.

Hinweise:

- Verwenden Sie die Methode `drawForward` der Klasse `Staffelei` um einen Strich zu malen. Als Länge eignen sich 10 Pixel.

- Rotationen können Sie mit der Methode `rotate` der Klasse `Staffelei` erreichen, die eine Gradzahl als Parameter bekommt.
- Unterscheiden Sie in Ihrer Methode `kochkurve` zwei Fälle: Entweder ist die Ordnung 0 (oder kleiner) oder die Ordnung ist mindestens 1. Im zweiten Fall soll die Methode `kochkurve` rekursiv verwendet werden.