Prof. Dr. J. Giesl

C. Aschermann, F. Frohn, J. Hensel, T. Ströder

Allgemeine Hinweise:

- Die **Hausaufgaben** sollen in Gruppen von je **2 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 Mittwoch, den 26.11.2014 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 Mittwoch, dem 26.11.2014 um 15:00 Uhr an Ihre Tutorin/Ihren Tutor.
  Stellen Sie sicher, dass Ihr Programm von `javac` **akzeptiert** wird, ansonsten werden keine Punkte vergeben.
- Aufgaben, die mit einem * markiert sind, sind Sonderaufgaben mit erhöhtem Schwierigkeitsgrad. Sie tragen nicht zur Summe der erreichbaren Punkte bei, die für die Klausurzulassung relevant ist, jedoch werden Ihnen die in solchen Aufgaben erreichten Punkte ganz normal gutgeschrieben.

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): (10 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 Integer i1;

    private int i2;

    private double d;

    private Float f;

    public B(int a, int b, int c, int d) {
        this.i1 = a;
        this.i2 = b;
        this.d = d;
        this.f = (float)c;
    }

    public B(Integer a, int b, double c, Float d) {
        this.i1 = a;
        this.i2 = b;
        this.d = c;
        this.f = d;
    }

    public int f(Float x, int y) {
        return 11;
    }

    public int f(double x, int y) {
        return 12;
    }
```

```
    public Integer f(double x, Long y) {
        return 13;
    }

    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.d);
        System.out.println(b1.f(5,6));
        System.out.println(b1.f(7f,8));
        System.out.println(b1.f(10f,17L));
        B b2 = new B(b1.i1, 5, 6, 9);
        System.out.println(b2.f);
        System.out.println(b2.f(b1.f,b1.i2));
        B b3 = new B(b2.i1, 14, 15, 16f);
        System.out.println(b3.d);
        System.out.println(b3.g(new Long(18)));
        System.out.println(b3.g(b1.i1));
        System.out.println(b3.f(b2.g(19), 21));
    }

}
```

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

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 (Einfache Rekursion): (4 Punkte)

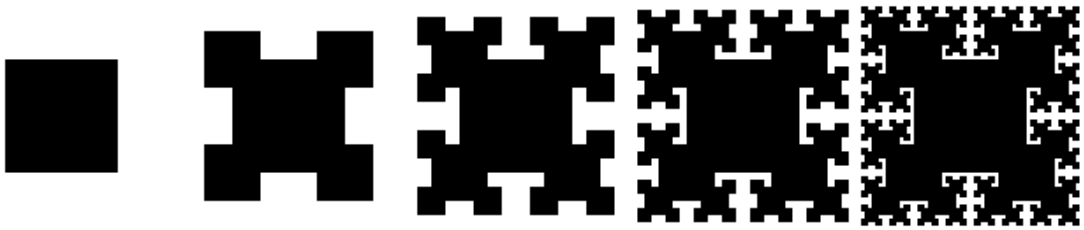
Betrachten Sie folgende Methode:

```
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 (Fraktale):

In dieser Aufgabe geht es darum, eine fraktale Struktur grafisch darzustellen. Die Struktur besteht aus Quadraten, die nach einem bestimmten Vorgehen angeordnet werden. Das Vorgehen lässt sich rekursiv beschreiben: Im ersten Level besteht die Struktur aus nur einem Quadrat. Die Struktur des n-ten Levels entsteht dadurch, dass erst ein Quadrat und danach auf dessen vier Ecken jeweils eine verkleinerte Variante des $(n-1)$-ten Levels der Struktur gezeichnet wird. Hierbei haben die größten Quadrate des $(n-1)$-ten Levels die halbe Kantenlänge des größten Quadrates des n-ten Levels. In folgendem Bild sind die ersten fünf Level der fraktalen Struktur dargestellt.

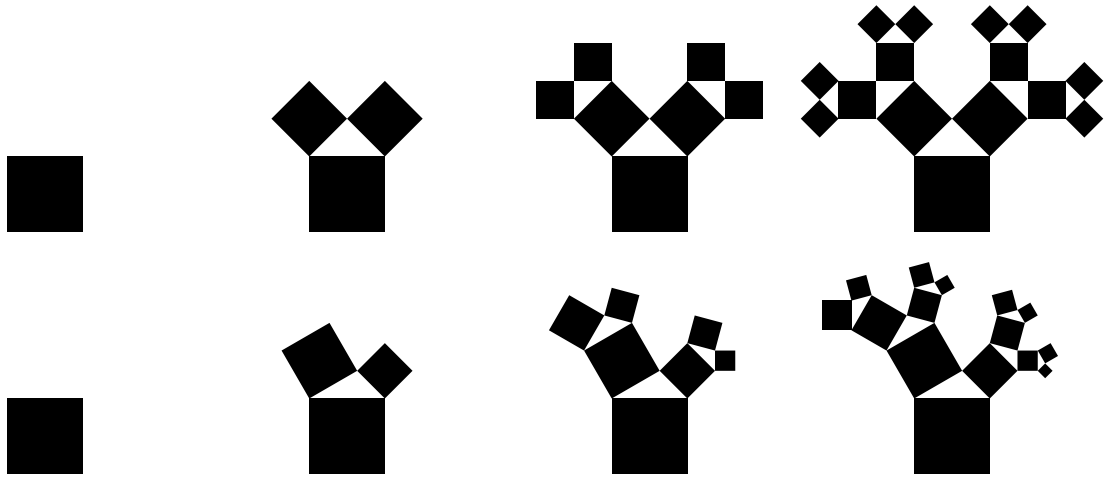


Verwenden Sie zur graphischen Darstellung die Klasse `Canvas`, welche in der Datei `Canvas.java` definiert ist. Sobald ein Objekt der Klasse `Canvas` erzeugt wird, wird dieses in einem Fenster auf dem Bildschirm angezeigt. Die Klasse `Canvas` stellt (unter anderem) die Methoden `move` und `square` zur Verfügung, die ein bzw. zwei `double`-Zahlen als Parameter bekommen und Zeichenoperationen in dem angezeigten Fenster ausführen. Sie können die einzelnen Zeichenschritte über die Schaltflächen Vor und Zurueck nacheinander ansehen. Die Schaltflächen `Anfang` und `Ende` zeigen die leere Fläche bzw. das fertige Bild.
Sei `c` ein Objekt der Klasse `Canvas`. Dann bewegt beispielsweise der Aufruf `c.move(3, 4)` die aktuelle Zeichenposition (d. h. die Position, an der als nächstes gezeichnet wird) um 3 Einheiten nach rechts und 4 Einheiten nach unten. Ein Aufruf `c.square(5)` zeichnet ein gefülltes Quadrat mit einer Seitenlänge von 5 Einheiten, wobei der Mittelpunkt des Quadrats auf der aktuellen Zeichenposition liegt.
Implementieren Sie die statische Methode `paintFractal` in der Klasse `Squares`, welche eine Referenz `c` auf ein Objekt der Klasse `Canvas`, eine `int`-Zahl `level` und eine `double`-Zahl `length` übergeben bekommt und mit Hilfe des `Canvas` Objekts `c` die beschriebene fraktale Struktur des `level`-ten Level zeichnet, wobei die Kantenlänge des größten Quadrats `length` beträgt. Sie dürfen in dieser Aufgabe keine Schleifen verwenden. Die Verwendung von Rekursion ist hingegen erlaubt.
Der folgende Aufruf eignet sich dazu, Ihr Programm zu testen.

```
java Squares 5 100
```

Aufgabe 6 (Fraktale): (15 + 4* Punkte)

Auch in dieser Aufgabe soll eine fraktale Struktur mithilfe der Klasse `Canvas` gezeichnet werden. Diesmal geht es um sogenannte Pythagoras Bäume. Ein Pythagoras Baum im ersten Level ist wiederum nur ein Quadrat einer vorgegebenen Basislänge. Ein Pythagoras Baum des n-ten Levels besteht aus einem Quadrat und zwei Pythagoras Unterbäumen des $(n-1)$-ten Levels. Die jeweils ersten Quadrate dieser beiden Unterbäume werden oberhalb des Quadrates im n-ten Level nebeneinander positioniert. Dabei wird das linke Quadrat um einen Winkel α gegen den Uhrzeigersinn und das rechte Quadrat um einen Winkel β im Uhrzeigersinn rotiert, sodass die drei Quadrate zwischen sich ein leeres Dreieck formen. Beide Winkel müssen positiv und kleiner als 90° sein und dürfen in Summe nicht mehr als 120° ergeben. Basierend auf der Länge des Quadrates im n-ten Level und diesen beiden Winkeln müssen die Längen der beiden Quadrate in den beiden Unterbäumen berechnet werden. Die folgende Grafik zeigt die ersten vier Level eines Pythagoras Baums mit den Winkeln $\alpha = 45°$ und $\beta = 45°$ (oben) sowie $\alpha = 30°$ und $\beta = 45°$ (unten).



Zur Berechnung der jeweils nächsten Quadratlängen eignet sich der Sinussatz. Sei ℓ die Länge des unteren Quadrats. Dann folgt aus dem Sinussatz, dass die Länge des linken Quadrats genau $\frac{\sin(\beta)\cdot\ell}{\sin(180°-\alpha-\beta)}$ und die Länge des rechten Quadrats genau $\frac{\sin(\alpha)\cdot\ell}{\sin(180°-\alpha-\beta)}$ ist. Um den Sinus eines Winkels `angle` (angegeben in Grad) in Java zu berechnen, können Sie den Ausdruck `Math.sin(Math.toRadians(angle))` verwenden.
Die Klasse `Canvas` bietet neben den bereits aus der vorigen Aufgabe bekannten Methoden `move` und `square` zusätzlich eine Methode `rotate` an, welche einen Winkel in Grad übergeben bekommt. Diese Methode rotiert die Ausrichtung für alle weiteren Zeichenoperationen um den übergebenen Winkel im Uhrzeigersinn (bei negativen Winkeln entsprechend gegen den Uhrzeigersinn).
Sei `c` ein Objekt der Klasse `Canvas`. Dann zeichnen beispielsweise die aufeinander folgenden Aufrufe `c.rotate(45)` und `c.square(5)` ein um 45° rotiertes Quadrat mit Seitenlänge 5 mit der aktuellen Position als Mittelpunkt. Analog wird die aktuelle Position durch die aufeinander folgenden Aufrufe `c.rotate(90)` und `c.move(1,0)` um eine Einheit nach unten statt nach rechts verschoben. Durch `rotate` wird also das gesamte Koordinatensystem rotiert.
Außerdem bietet die Klasse `Canvas` zwei Farben `BROWN` und `GREEN` als statische Attribute an, welche mittels der Methode `chooseColor` ausgewählt werden können. Der Aufruf `c.chooseColor(Canvas.BROWN)` führt also im obigen Beispiel dazu, dass alle weiteren Zeichenoperationen mit brauner Farbe durchgeführt werden. Diese Farben sollen dazu genutzt werden, große Quadrate braun und kleine Quadrate grün zu färben, damit die grafische Qualität der Pythagoras Bäume erhöht wird.
Wie in der vorigen Aufgabe dürfen Sie in dieser Aufgabe keine Schleifen verwenden. Die Verwendung von Rekursion ist hingegen erlaubt.

a) Implementieren Sie die statische Methode `paintPythagorasTree` in der Klasse `Pythagoras`, welche folgende Parameter erhält:

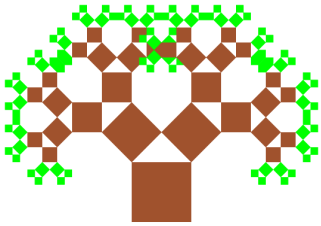
- eine Referenz `c` auf ein `Canvas` Objekt
- eine `int`-Zahl `level`, welche das gewünschte Level des Pythagoras Baums angibt
- einen `double`-Wert `length`, welcher die Länge des ersten Quadrats des Pythagoras Baums angibt

- eine `int`-Zahl `leftAngle`, welche dem Winkel α entspricht
- eine `int`-Zahl `rightAngle`, welche dem Winkel β entspricht
- eine `int`-Zahl `switchLength`, welche die maximale Länge von Quadraten angibt, welche grün gezeichnet werden sollen

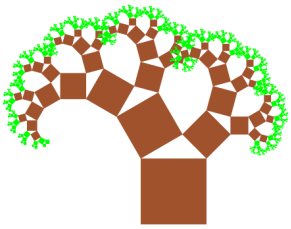
Diese Methode soll einen Pythagoras Baum des spezifizierten Levels zeichnen, wobei das erste Quadrat die übergebene Länge hat und die beiden jeweils folgenden Quadrate wie oben beschrieben um die Winkel α und β rotiert werden. Außerdem sollen Quadrate mit einer Seitenlänge größer als `switchLength` in braun und alle anderen Quadrate in grün gezeichnet werden.

Zum Testen Ihrer Implementierung eignen sich die folgenden Aufrufe Ihres Programms (daneben finden Sie Abbildungen, die Sie als Ergebnis zu diesen Aufrufen erhalten sollten):

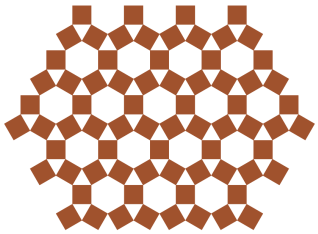
- `java Pythagoras 7 100 45 45 20`



- `java Pythagoras 10 100 30 45 10`

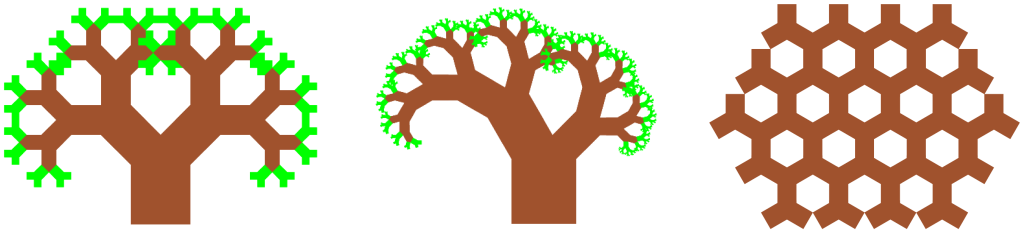


- `java Pythagoras 7 50 60 60`



b)* Um die grafische Qualität weiter zu verbessern, sollten die leeren Dreiecke zwischen den Quadraten gefüllt werden. Modifizieren Sie die Methode `paintPythagorasTree` so, dass sie dieses Verhalten zeigt (falls Sie diese Teilaufgabe lösen, genügt natürlich die Abgabe der Version mit gefüllten Dreiecken zur Lösung beider Teilaufgaben). Nutzen Sie zum Zeichnen lediglich die bereits von der Klasse `Canvas` bereitgestellten Methoden.

Die drei Testaufrufe aus der vorigen Teilaufgabe liefern dann folgende Bilder:



Hinweise:

- Die Höhe eines beliebigen Dreiecks von der Hypothenuse mit Länge c aus berechnet sich durch $\frac{\sqrt{2(a^2b^2+a^2c^2+b^2c^2)-(a^4+b^4+c^4)}}{2c}$, wobei a und b die Längen der beiden Katheten des Dreiecks sind. Die Wurzel lässt sich in Java mittels der Methode `Math.sqrt` ziehen.