Prof. Dr. J. Giesl
D. Cloerkes, S. Dollase, N. Lommen, D. Meier, F. Meyer

Allgemeine Hinweise:

- Die **Deadline** zur **Abgabe** der Hausaufgaben ist am **Donnerstag, den 25.11.2021, um 18:00 Uhr**.
- Der **Workflow** sieht wie folgt aus. Die Abgabe der Hausaufgaben erfolgt **im Moodle-Lernraum** und kann nur in **Zweiergruppen** stattfinden. Dabei müssen die Abgabepartner*innen **dasselbe Tutorium** besuchen. Nutzen Sie ggf. das entsprechende **Forum** im Moodle-Lernraum, um eine*n Abgabepartner*in zu finden. Es darf **nur ein*e** Abgabepartner*in die Abgabe hochladen. Diese*r muss sowohl die **Lösung** als auch den **Quellcode** der Programmieraufgaben hochladen. Die Bepunktung wird dann von uns für **beide** Abgabepartner*innen **separat** im Lernraum eingetragen. Die Feedbackdatei ist jedoch nur dort sichtbar, wo die Abgabe hochgeladen wurde und muss innerhalb des Abgabepaars **weitergeleitet** werden.
- Die **Lösung** muss als PDF-Datei hochgeladen werden. Damit die Punkte beiden Abgabepartner*innen zugeordnet werden können, müssen **oben** auf der **ersten Seite** Ihrer Lösung die **Namen**, die **Matrikelnummern** sowie die **Nummer des Tutoriums** von **beiden** Abgabepartner*innen angegeben sein.
- Der **Quellcode** der Programmieraufgaben muss als `.zip`-Datei hochgeladen werden und **zusätzlich** in der PDF-Datei mit Ihrer Lösung enthalten sein, sodass unsere Hiwis ihn mit Feedback versehen können.

  Auf diesem Blatt muss Ihre Codeabgabe Ihren vollständigen Java-Code in Form von `.java`-Dateien enthalten. Aus dem Lernraum heruntergeladene Klassen, etwa die Datei `SimpleIO.java`, dürfen nicht mit abgegeben werden.

  Stellen Sie sicher, dass Ihr Programm von `javac` **akzeptiert** wird, wenn die entsprechenden Klassen aus dem Lernraum hinzugefügt werden. Ansonsten werden keine Punkte vergeben.
- Einige Hausaufgaben müssen im Spiel Codescape gelöst werden. Klicken Sie dazu im Lernraum rechts im Block "Codescape" auf den angegebenen Link. Diese Aufgaben werden getrennt von den anderen Hausaufgaben gewertet.

Tutoraufgabe 1 (Überblickswissen):

a) Was ist der Unterschied zwischen `int` und `Integer` in Java?

b) Was ist ein `StackOverflowError` in Java und was ist der häufigste Grund dafür?

Tutoraufgabe 2 (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:

Listing 1: A.java

```
1  public class A {
2      public static void main(String[] args) {
3          A a = new A();
4
5          a.a(Long.valueOf(100)); //a)
6          a.a(Double.valueOf(100)); //b)
7          a.a(Integer.valueOf(100)); //c)
8
9          b(Integer.valueOf(100), "0"); //d)
10         b(100L, "0"); //e)
```

```
11          b(100L, '0'); //f)
12      }
13
14      public void a(int p) {
15          System.out.println("a1");
16      }
17
18      public void a(double p) {
19          System.out.println("a2");
20      }
21
22      public void a(Double p) {
23          System.out.println("a3");
24      }
25
26
27      public static void b(Long p1, int p2) {
28          System.out.println("b1");
29      }
30
31      public static void b(long p1, String p2) {
32          System.out.println("b2");
33      }
34
35      public static void b(Long p1, String p2) {
36          System.out.println("b3");
37      }
38  }
```

Geben Sie die Ausgabe dieses Programms an, wenn die `main`-Methode ausgeführt wird. **Begründen Sie Ihre Antwort!** Ordnen Sie jeder Teilaufgabe die aufgetretenen Effekte zu und erklären Sie, warum gerade diese zu beobachten sind.

Tutoraufgabe 3 (Programmanalyse (Video)):

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:

Listing 2: A.java

```
1   public class A {
2       private Integer i;
3       private double d;
4
5       public A() {
6           this.i = 1;
7           this.d = 4;
8       }
9
10      public A(Integer x, double y) {
11          this.i = x;
12          this.d = y;
13      }
14
15      public A(int x, double y) {
16          this.i = 3;
17          this.d = x + y;
18      }
19
20      public int f(Integer x) {
```

```
21          return this.i + x;
22      }
23
24      public int f(double i) {
25          this.d = i;
26          return this.i;
27      }
28
29      public static void main(String[] args) {
30          A a1 = new A();
31          System.out.println(a1.f(5));                    // a)
32          System.out.println(a1.d);                       // b)
33          System.out.println(a1.f(Long.valueOf(2)));      // c)
34          A a2 = new A(1,1);
35          System.out.println(a2.i);                       // d)
36          System.out.println(a2.d);                       // e)
37      }
38  }
```

Geben Sie die Ausgabe dieses Programms an, wenn die `main`-Methode ausgeführt wird. **Begründen Sie Ihre Antwort!** Ordnen Sie jeder Teilaufgabe die aufgetretenen Effekte zu und erklären Sie, warum gerade diese zu beobachten sind. Nehmen Sie dabei auch Bezug auf die Konstruktor-Aufrufe.

Aufgabe 4 (Programmanalyse): (2+2+2+2+2+2+2+2+2+2 = 20 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:

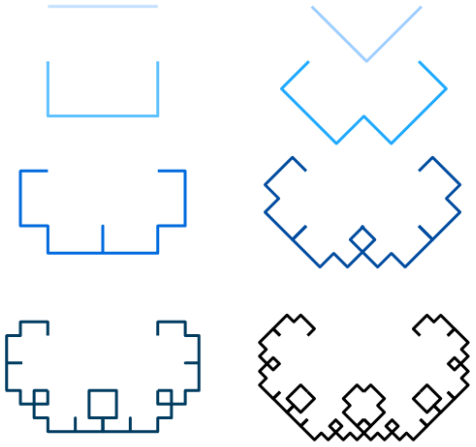
```
public class B {

    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 = (double)d;
        this.f = c;
    }

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

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

    public int f(int x, float y) {
```

```
        return 12;
    }

    public int f(Double x, long y) {
        return 13;
    }

    public double g(Float x) {
        return 7.0;
    }

    public Float g(double x) {
        return 8f;
    }

    public static void main(String[] args) {
        B b1 = new B(1,2,3,4);
        System.out.println(b1.d);                          // a)
        System.out.println(b1.f(7d,8L));                   // b)
        System.out.println(b1.f(10d,17));                  // c)
        System.out.println(b1.f(5,6L));                    // d)
        B b2 = new B(b1.i1, 5, 6, 9);
        System.out.println(b2.f);                          // e)
        System.out.println(b2.f(b1.f,b1.i2));              // f)
        B b3 = new B(b2.i1, 14, 1.5, 16);
        System.out.println(b3.d);                          // g)
        System.out.println(b3.g(b1.i1));                   // h)
        System.out.println(b3.g(Float.valueOf(18)));       // i)
        System.out.println(b3.f(b2.g(19f), 21));           // j)
    }

}
```

Geben Sie die Ausgabe dieses Programms an, wenn die `main`-Methode ausgeführt wird. **Begründen Sie Ihre Antwort!** Ordnen Sie jeder Teilaufgabe die aufgetretenen Effekte zu und erklären Sie, warum gerade diese zu beobachten sind. Nehmen Sie dabei auch Bezug auf die Konstruktor-Aufrufe.

Tutoraufgabe 5 (Rekursion):

In dieser Aufgabe soll die fraktale Struktur der Lévy–C–Kurven mithilfe der Klasse `Canvas` gezeichnet werden, die im Moodle zusammen mit den anderen Aufgabendokumenten bereitgestellt ist. Verwenden Sie das Programm `javadoc`, um die Schnittstellendokumentation dieser Klasse zu erzeugen. In dieser Aufgabe sind die Methoden `rotate` und `drawForward` relevant.
Eine Lévy–C–Kurve 0. Ordnung besteht aus einer geraden Linie. Kurven i-ter Ordnung werden gebildet, indem man zunächst eine Lévy–C–Kurve $(i-1)$-ter Ordnung zeichnet. Vom letzten Strich dieser Kurve aus zeichnet man dann in einem Winkel von $90° - 90° \cdot (i-1)$ eine weitere Lévy–C–Kurve $(i-1)$-ter Ordnung. Ein Winkel von $0°$ bedeutet hierbei, dass die letzte Linie der Kurve gerade in die erste Linie der zweiten Kurve über geht. Positive Winkel bedeuten eine Ecke im Uhrzeigersinn, negative Winkel eine Ecke entgegen dem Uhrzeigersinn. Die folgende Grafik zeigt Lévy–C-Kurven der Ordnungen 0 bis 7. In ihrer Implementierung werden die Kurven zum Teil *anders gedreht* sein.



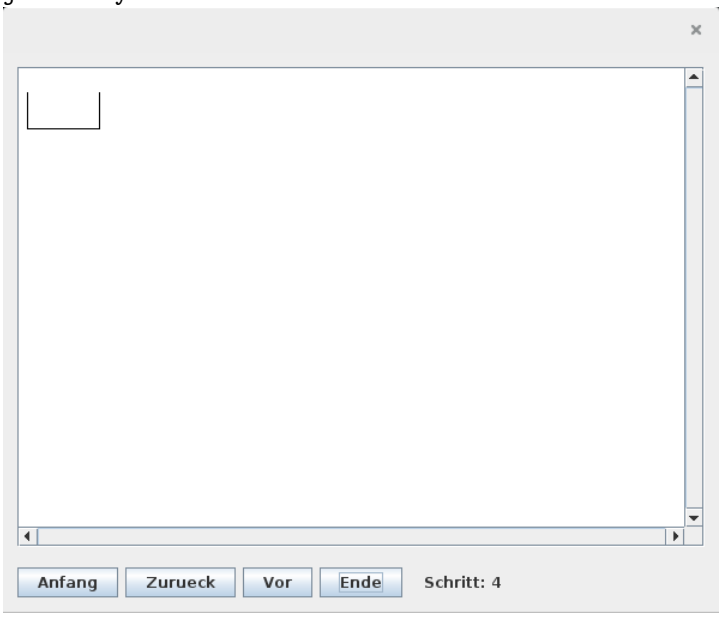
[1]

Sie dürfen in dieser Aufgabe keine Schleifen verwenden. Die Verwendung von Rekursion ist hingegen erlaubt. Implementieren Sie die statische Methode `levyCKurve` in der Klasse `LevyC`, welche folgende Parameter erhält:

- eine Referenz `c` auf ein `Canvas` Objekt
- eine `int`-Zahl, welche die gewünschte Ordnung der Kurve angibt.
- eine `int`-Zahl, welche die Länge einer Lévy–C-Kurve 0. Ordnung angibt.

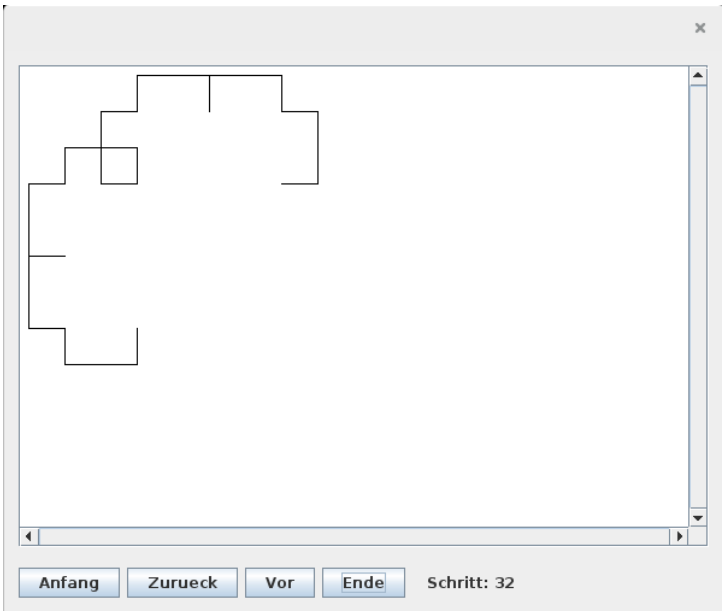
Diese Methode soll eine Lévy–C-Kurve der spezifizierten Ordnung zeichnen.
Zum Testen Ihrer Implementierung enthält die Klasse `LevyC` schon eine `main`-Methode. Das Programm bekommt bis zu zwei Parameter. Der erste gibt die Ordnung der Kurve an, der zweite die Länge der Kurven 0. Ordnung, aus denen sie zusammengesetzt werden soll. Aus der `main`-Methode wird die Methode `levyCKurve` entsprechend aufgerufen. Sie können ihre Implementierung mit folgenden Aufrufen testen (darunter finden Sie Abbildungen, die Sie als Ergebnis zu diesen Aufrufen erhalten sollten):

- `java LevyC 2 30`

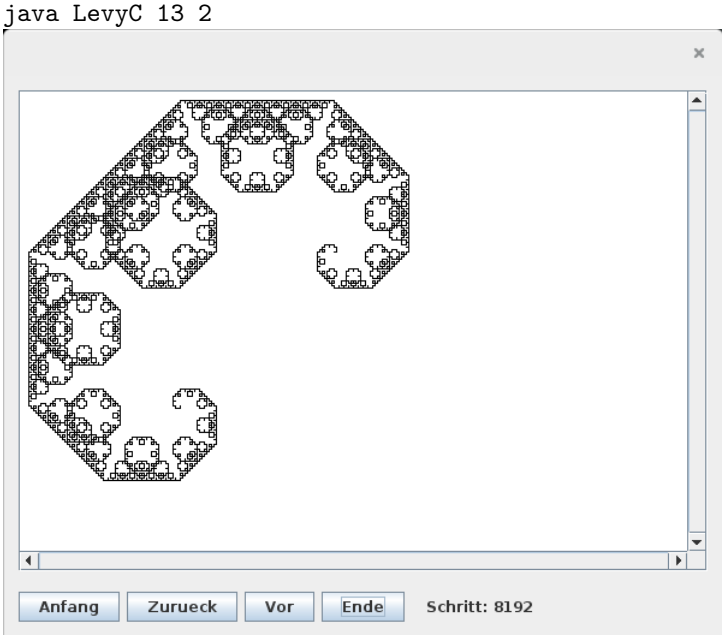


- `java LevyC 5 30`

[1] Bild lizenziert unter CC BY-SA 3.0, Autor: Gandalf61 at English Wikipedia, Quelle: `https://en.wikipedia.org/wiki/File:Levy_C_construction.png`



- `java LevyC 13 2`



Hinweise:

- Implementieren Sie die Methode `LevyC` rekursiv.
- Klicken Sie einmal auf die Schaltfläche Ende, um das Ergebnis anzuzeigen.
- Mit den Schaltflächen Vor und Zurueck können Sie die Zeichnung schrittweise auf- bzw. abbauen. Der Ablauf entspricht dabei dem Ablauf Ihres Programms. Die Schaltflächen Anfang und Ende springen zum Anfang bzw. Ende des Ablaufs.

Tutoraufgabe 6 (Rekursion (Video)):

Die Fibonacci-Zahlen sind wie folgt definiert: $F_0 = 0$, $F_1 = 1$ und $F_n = F_{n-1} + F_{n-2}$ für $n > 1$. Schreiben Sie eine Klasse `Fibonacci`, welche die zwei statischen Methoden `calculateIterative` und `calculateRecursive` enthält. Diese Methoden erhalten jeweils einen `int`-Parameter n und geben die n-te Fibonacci-Zahl zurück.

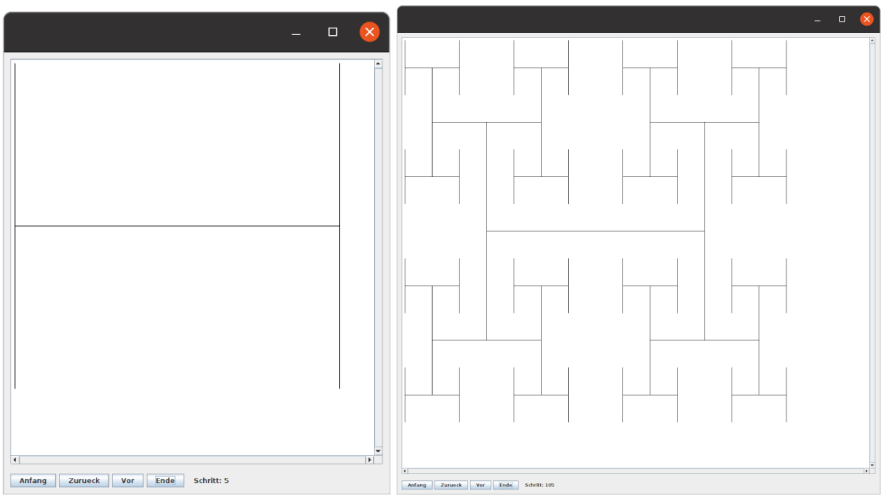


Abbildung 1: Beispiel: Programmausgabe für einen H-Baum der Tiefe 1 (links) und 3 (rechts)

Dabei soll die erstgenannte Methode keine Rekursion und die zweitgenannte Methode keine Schleifen (aber Rekursion) benutzen. Die rekursive Methode soll dabei nicht mehr als n rekursive Aufrufe brauchen, um die n-te Fibonacci-Zahl zu berechnen.

Aufgabe 7 (Rekursion): (30 Punkte)

Auch in dieser Aufgabe soll eine fraktale Struktur mithilfe der Klasse `Canvas` gezeichnet werden. Hier geht es um sogenannte H-Bäume.[2] Dabei geht es um ein Fraktal, das wie folgt entsteht. Ein H-Baum der Tiefe 1 ist lediglich ein H. Ein H-Baum der Tiefe n ist ein H, bei dem an den vier Enden des Hs je ein H-Baum der Tiefe $n-1$ hängt, so dass die Mitte des Mittelstrichs des größten Hs des Teilbaums der Tiefe $n-1$ das Ende des aktuellen Hs berührt. Eine Ausgabe des Programms könnte für die Tiefen 1 und 3 wie in Abbildung 1 dargestellt aussehen.
Ihre Aufgabe ist, eine Methode `drawHTree(int size, int n)` zu implementieren, die solche Bäume zeichnet. Der Parameter `size` bestimmt die Größe des größten Hs des Baums, wobei die Größe die Länge der H-Striche angibt (anders als in den meisten Schriftarten soll hier der Mittelstrich des Hs genauso lang sein wie die Seitenstriche). Der Parameter `n` bestimmt die Tiefe des zu zeichnenden Baums. Ein einfaches Gerüst, um Ihre Implementierung zu testen, ist in `HTree.java` gegeben. Ändern Sie die Parameter des `drawHTree`-Aufrufs in der `main`-Methode nach Belieben um das Programm auf Richtigkeit zu prüfen. Die Größe der Bäume sollte in jedem Rekursionsschritt halbiert werden. Benutzen Sie für die Implementierung keine Schleifen, sondern nur Rekursion.
Die Klasse `Canvas` bietet neben den bereits aus Aufgabe 5 bekannten Methoden `rotate` und `drawForward` zusätzliche Methoden an. In dieser Aufgabe wird zusätzlich noch die Methode `moveForward` relevant, die genau wie `drawForward` die aktuelle Position verändert, jedoch nichts zeichnet. Außerdem wird das Methodenpaar `push` und `pop` bereitgestellt. Die Methode `push` speichert die momentane Konfiguration (Ausrichtung und Position des Zeigers) auf einem Stack, während `pop` die oberste auf dem Stack liegende Konfiguration wieder herstellt. Als Beispiel für die Funktionsweise sehen Sie unten eine beispielhafte Verwendung von `push` und `pop` für ein Objekt `c` vom Typ `Canvas`, wobei die Kommentare die jeweils aktuelle Zeigerposition angeben. Neu erstellte `Canvas`-Objekte `c` sind stets nach unten ausgerichtet.

```
1  //Position: x:0,y:0 Ausrichtung: 0 Grad (nach unten)
2  c.push();
3  c.moveForward(10);
4  //Position: x:0,y:10 Ausrichtung: 0 Grad (nach unten)
5  c.push()
6  c.moveForward(10);
```

[2] https://de.wikipedia.org/wiki/H-Baum

```
7   c.rotate(180);
8   //Position: x:0,y:20 Ausrichtung: 180 Grad (nach oben)
9   c.pop()
10  //Position: x:0,y:10 Ausrichtung: 0 Grad (nach unten)
11  c.pop()
12  //Position: x:0,y:0 Ausrichtung: 0 Grad (nach unten)
```

Aufgabe 8 (Deck 5): (Codescape)

Lösen Sie die Missionen von Deck 5 des Codescape Spiels. Ihre Lösung für die Codescape Missionen wird nur dann für die Zulassung gezählt, wenn Sie Ihre Lösung vor der einheitlichen Codescape Deadline am Samstag, den 22.01.2022, um 23:59 Uhr abschicken.