

Tutoraufgabe 1 (Überblickswissen):

- Was ist der Unterschied zwischen `int` und `Integer` in Java?
- Was ist ein `StackOverflowError` in Java und was ist der häufigste Grund dafür?

Lösung: _____

- Der Typ `int` ist ein primitiver Typ, der Typ `Integer` ist nicht primitiv, sondern die zu `int` gehörende Hüllklasse. Nur `Integer`-Variablen können den Wert `null` annehmen, bei `int`-Variablen ist dies hingegen nicht möglich. Normalerweise sollte immer mit dem primitiven Typ `int` gearbeitet werden. Manche Funktionen arbeiten jedoch ausschließlich auf Objekten, dann wird die Hüllklasse benötigt.
- Ein `StackOverflowError` ist ein Fehler, der auftritt, wenn die Größe des Laufzeitkellerspeichers die den von Java dafür allozierten Speicher übertrifft. Der häufigste Grund dafür ist entweder sehr tiefe oder nicht terminierende Rekursion.

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.

Lösung: _____

- Die erste Ausgabe ist **a2**, da der Parameter vom Typ `Long` durch Unboxing zu `long` und anschließend durch implizite Typumwandlung zu `double` wird.
- Die zweite Ausgabe ist **a3**, da die Signatur der Methode genau passt.
- Die dritte Ausgabe ist **a1**, da der Parameter vom Typ `Integer` durch Unboxing zu `int` wird.
- Die vierte Ausgabe ist **b2**, da der erste Parameter vom Typ `Integer` durch Unboxing zu `int` und anschließend durch implizite Typumwandlung zu `long` wird.
- Die fünfte Ausgabe ist **b2**, da die Signatur der Methode genau passt.
- Die sechste Ausgabe ist **b1**, da der zweite Parameter vom Typ `char` durch implizite Typumwandlung zu `int` wird.

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.

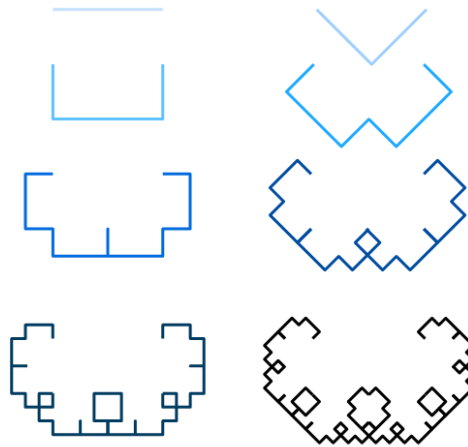
Lösung: _____

- Die erste Ausgabe ist 1. Da die implizite Typanpassung Vorrang vor dem Autoboxing hat, wird die zweite `f` Methode ausgeführt. Der Rückgabewert dieser Methode ist der Wert des `i` Attributs, welcher im ersten Konstruktor auf 1 gesetzt wurde.
- Die zweite Ausgabe ist 5.0. Bei der Ausführung der zweiten `f` Methode wurde das `d` Attribut auf den übergebenen Wert 5.0 gesetzt.
- Die dritte Ausgabe ist wieder 1. Hier wird zunächst das `Long` Objekt durch Unboxing in einen `long` Wert umgewandelt, der anschließend durch implizite Typanpassung in einen `double` Wert konvertiert wird. Also wird wieder die zweite `f` Methode ausgeführt und das (unveränderte) `i` Attribut ausgegeben.
- Die vierte Ausgabe ist 3. Dem Konstruktor werden zwei `int` Werte übergeben, sodass der dritte Konstruktor ausgeführt wird, da der zweite Konstruktor gegenüber dem dritten ein zusätzliches Autoboxing erfordern würde. Dieser belegt das `i` Attribut mit 3.
- Die fünfte Ausgabe ist 2.0. Auf dem für die letzte Ausgabe beschriebenen Ausführungsweg wurde der dritte Konstruktor mit den `int` Werten 1 und 1 aufgerufen. Deren Summe wird durch implizite Typanpassung zu 2.0 konvertiert und dem `d` Attribut zugewiesen.

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 $90^\circ - 90^\circ \cdot (i - 1)$ 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

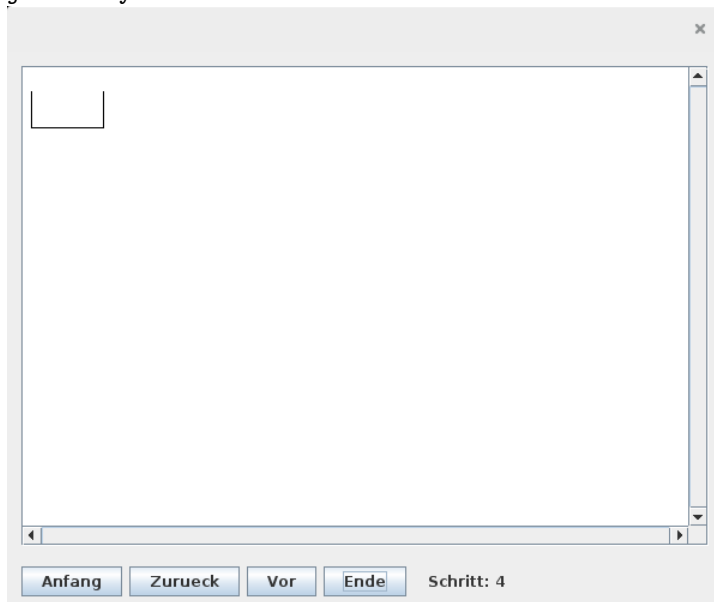
Sie dürfen in dieser Aufgabe keine Schleifen verwenden. Die Verwendung von Rekursion ist hingegen erlaubt. Implementieren Sie die statische Methode `levyCKurve` in der Klasse `LevyC`, welche folgende Parameter erhält:

- 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.

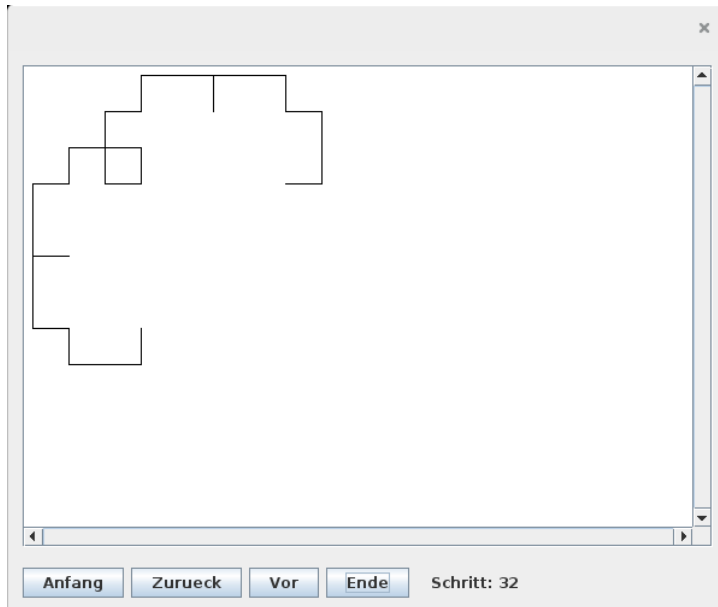
Zum Testen Ihrer Implementierung enthält die Klasse `LevyC` schon eine `main`-Methode. Das Programm bekommt bis zu zwei Parameter. Der erste gibt die Ordnung der Kurve an, der zweite die Länge der Kurven 0. Ordnung, aus denen sie zusammengesetzt werden soll. Aus der `main`-Methode wird die Methode `levyCKurve` 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`

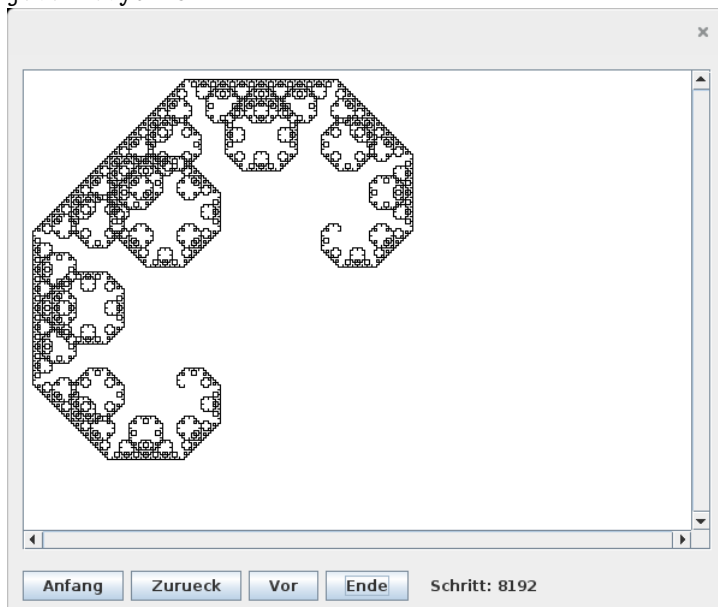


- `java LevyC 5 30`

¹Bild 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.

Lösung: _____

Listing 3: `LevyC.java`

```
public class LevyC {
```

```

static void levyCKurve(Canvas c, int ordnung, int length) {
    if(ordnung <= 0) {
        c.drawForward(length);
    } else {
        levyCKurve(c, ordnung-1, length);
        c.rotate(90 - 90* (ordnung-1));
        levyCKurve(c, ordnung-1, length);
    }
}

public static void main(String[] args) {
    int ordnung = 5;
    int length = 30;
    if(args.length==2) {
        length = Integer.parseInt(args[1]);
        ordnung = Integer.parseInt(args[0]);
    }
    else if(args.length==1) {
        ordnung = Integer.parseInt(args[0]);
    }
    else {
        System.out.println("Verwende Standardwerte: Ordnung 5, Laenge 30.");
        System.out.println("Verwendung: java LevyC Ordnung Laenge");
    }

    if (ordnung < 0) {
        System.out.println("Die Rekursionsordnung muss nicht-negativ sein!");
        return;
    }
    if (length < 1) {
        System.out.println("Die Laenge muss positiv sein!");
        return;
    }
    Canvas c = new Canvas();
    LevyC.levyCKurve(
        c,
        ordnung,
        length
    );
    c.refresh();
}
}

```

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. 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.

Lösung: _____

Listing 4: Fibonacci.java

```

public class Fibonacci{
    public static int calculateIterative(int n){

```

```

    int nextValue=1;
    int value=0;
    for(int i=0;i<n;++i){
        int temp=nextValue;
        nextValue=value+nextValue;
        value=temp;
    }
    return value;
}

public static int calculateRecursive(int n){
    return recursiveHelp(n,1,0);
}
private static int recursiveHelp(int n,int current,int prev){
    if(n==0){
        return prev;
    }
    return recursiveHelp(n-1, current+prev,current);
}

}

```