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

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?

Lösung:

a) 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.

b) 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)    "a2"
6          a.a(Double.valueOf(100)); //b)  "a3"
7          a.a(Integer.valueOf(100)); //c) "a1"
8
9          b(Integer.valueOf(100), "0"); //d)  "b2"
10         b(100L, "0"); //e)                  "b2"
11         b(100L, '0'); //f)
12     }
13
14     public void a(int p) {                  b1: 1x autoboxing, 1x implizite Typanpassung
15         System.out.println("a1");           b2: 5x implizite Typanpassung
16     }                                       b3: 1x autoboxing, 5x implizite Typanpassung
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:

a) Die erste Ausgabe ist `a2`, da der Parameter vom Typ `Long` durch Unboxing zu `long` und anschließend durch implizite Typumwandlung zu `double` wird.

b) Die zweite Ausgabe ist `a3`, da die Signatur der Methode genau passt.

c) Die dritte Ausgabe ist `a1`, da der Parameter vom Typ `Integer` durch Unboxing zu `int` wird.

d) Die vierte Ausgabe ist `b2`, da der erste Parameter vom Typ `Integer` durch Unboxing zu `int` und anschließend durch implizite Typumwandlung zu `long` wird.

e) Die fünfte Ausgabe ist `b2`, da die Signatur der Methode genau passt.

f) 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:

a) 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.

b) Die zweite Ausgabe ist `5.0`. Bei der Ausführung der zweiten `f` Methode wurde das `d` Attribut auf den übergebenen Wert `5.0` gesetzt.

c) 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.

d) 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.

e) 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.

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.

Lösung:

a) Die erste Ausgabe ist `4.0`. Der erste Konstruktor wird passend aufgerufen und dieser belegt das Attribut `d` mit dem implizit zu einem `Double` mit Autoboxing angepassten Wert `4.0`. Beachten Sie, dass das Programm ohne den expliziten Cast nicht kompilieren würde, da dann wiederum die implizite Typanpassung alleine nicht anwendbar wäre und Autoboxing ebenfalls versagen würde.

b) Die zweite Ausgabe ist `13`. Diese Methode ist die einzige, die ein `double` im ersten Parameter und ein `long` im zweiten Parameter verarbeiten kann, da weder `double` implizit nach `int`, noch `long` implizit nach `int` konvertiert werden kann.

c) Die dritte Ausgabe ist `11`, da die Parameter genau auf die Signatur der Methode passen.

d) Die vierte Ausgabe ist `12`, da diese Methode als einzige mit impliziter Typanpassung erreichbar ist.

e) Die fünfte Ausgabe ist `6.0`. Der einzige passende Konstruktor ist der erste, denn der zweite ist nicht anwendbar, da die implizite Typanpassung alleine nicht anwendbar ist und Autoboxing ebenfalls nicht zum Ziel für den zweiten Konstruktor führt. Das erste Argument wird durch Unboxing umgewandelt und das Attribut `f` mit dem dritten Parameter belegt, der implizit zu einem `float` Wert konvertiert wird.

f) Die sechste Ausgabe ist `11`. Die erste `f` Methode ist mit einer impliziten Typanpassung erreichbar und wird daher ausgeführt. Die dritte ist nicht anwendbar, da `float` nicht implizit zu `Double` umgewandelt werden kann.

g) Die siebte Ausgabe ist `1.5`. Diesmal ist der erste Konstruktor nicht anwendbar, da das dritte Argument nicht implizit von `double` zu `int` konvertiert werden kann. Also wird der zweite Konstruktor ausgeführt. Hierbei wird das vierte Argument über implizite Typanpassung konvertiert, während das dritte Argument über Autoboxing umgewandelt wird. Im dritten Konstruktor wird nun das Attribut `d` mit dem dritten Parameter belegt.

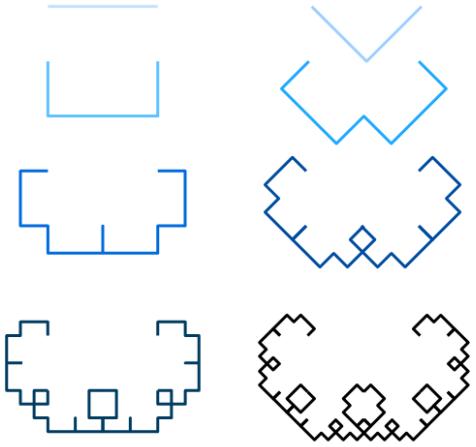
h) Die achte Ausgabe ist `8.0`, da `Integer` nicht zu `Float` umgewandelt werden kann, wohl aber nach Unboxing zu `double`.

i) Die neunte Ausgabe ist `7.0`. Das Argument ist ein `Float` Objekt und damit passt die erste `g` Methode ohne Anpassungen und wird ausgeführt.

j) Die zehnte Ausgabe ist `11`. Zunächst wird die zweite `g` Methode ausgeführt, da wiederum implizite Typanpassung Vorrang vor Autoboxing hat. Deren Ergebnis ist vom Typ `Float`. Damit passt die erste `f`-Methode nach Unboxing und einer impliziten Typumwandlung.

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.



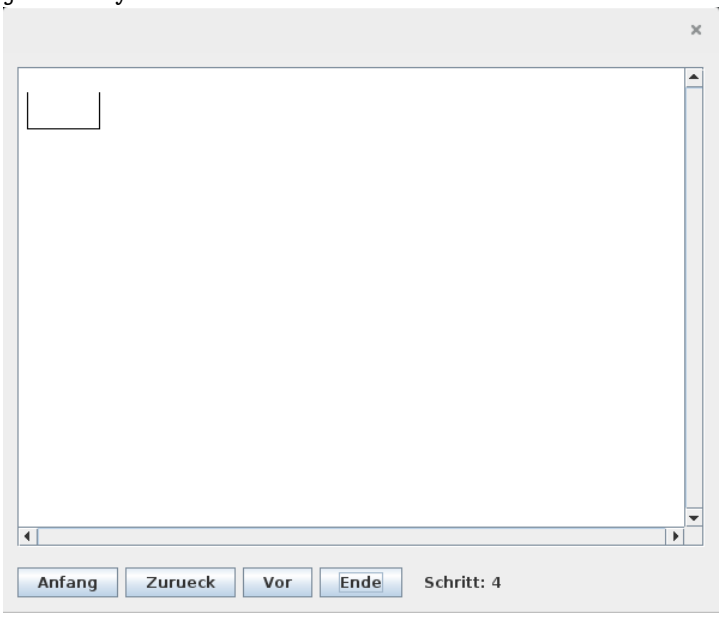
[1]

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.

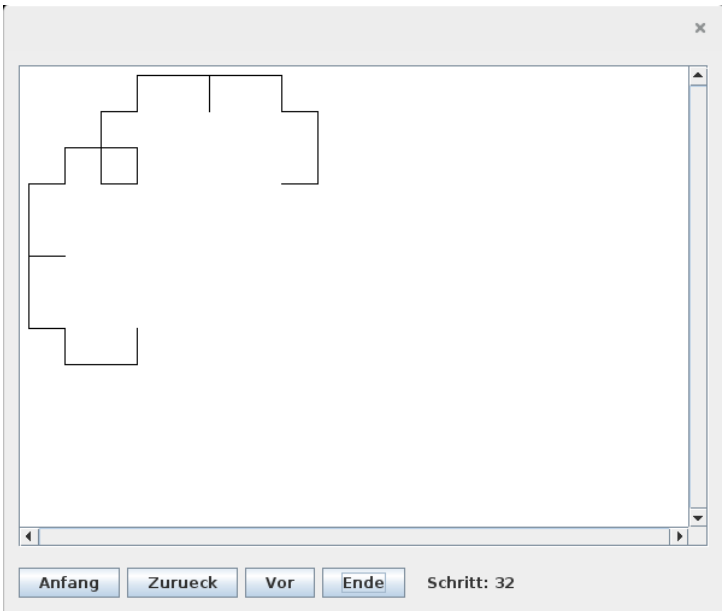
Diese Methode soll eine Lévy–C-Kurve der spezifizierten Ordnung zeichnen.
Zum Testen Ihrer Implementierung enthält die Klasse `LevyC` schon eine `main`-Methode. Das Programm bekommt bis zu zwei Parameter. Der erste gibt die Ordnung der Kurve an, der zweite die Länge der Kurven 0. Ordnung, aus denen sie 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`

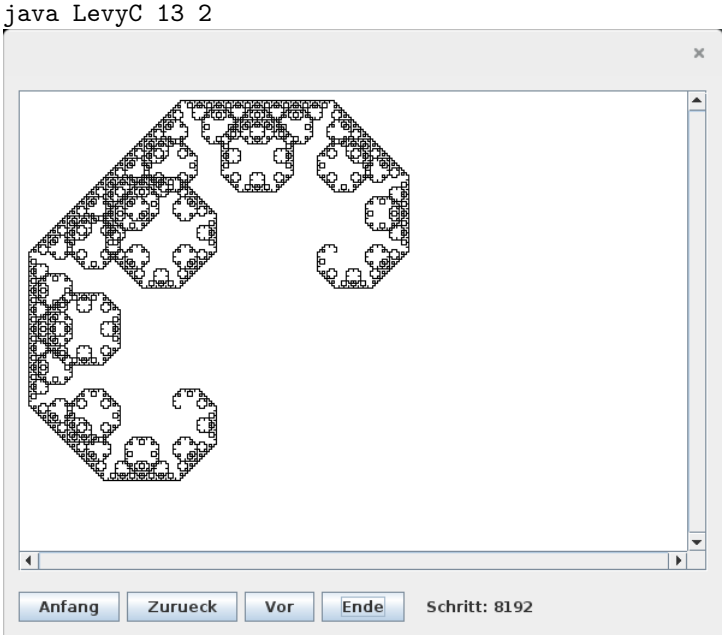


- `java LevyC 5 30`

[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.

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){
```

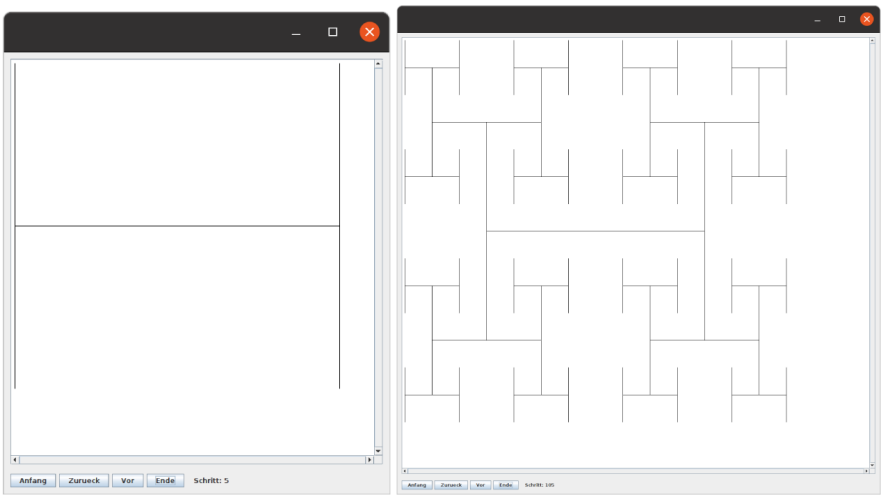


Abbildung 1: Beispiel: Programmausgabe für einen H-Baum der Tiefe 1 (links) und 3 (rechts)

```
    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);
  }


}
```

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

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

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);
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)
```

Lösung:

Listing 5: HTree.java

```
public class HTree {
    private Canvas c;

    public static void main(String[] args) {
        HTree t=new HTree();
        t.drawHTree(600,5);
    }

    public HTree(){
        c=new Canvas(); //Ausrichtung ist bei Neuerstellung nach unten
    }

    /*Diese Methode zeichnet einen H-Tree der Tiefe n. Vor dem Aufruf der Methode
    muss sichergestellt sein,
    dass die Canvas Zeichenrichtung nach unten gerichtet ist. */
    public void drawHTree(int size, int n) {
        if(n<=0) {
            return;
        }

        int hs=size/2;

        c.push();
        //Zur oberen linken Ecke des Hs bewegen
        c.rotate(90);
        c.moveForward(hs);
        c.rotate(90);
        c.moveForward(hs);
```

```
        //Zeichne einen H-Tree der Tiefe n-1 an der oberen linken Ecke.
        //Beachtet, dass vorher rotiert werden muss,
        //damit die Ausrichtung nach unten zeigt
        c.rotate(180);
        drawHTree(hs,n-1);

        //Linker Strich des momentan gezeichneten Hs
        c.drawForward(size);

        //Zeichne einen H-Tree der Tiefe n-1 an der unteren linken Ecke
        drawHTree(hs,n-1);

        //Zur Mitte des linken Striches bewegen
        c.rotate(180);
        c.moveForward(hs);

        //Mittelstrich des Hs zeichnen
        c.rotate(90);
        c.drawForward(size);

        //Zur oberen rechten Ecke des Hs bewegen
        c.rotate(270);
        c.moveForward(hs);

        //Zeichne einen H-Tree der Tiefe n-1 an der oberen rechten Ecke.
        c.rotate(180);
        drawHTree(hs,n-1);

        //Rechten Strich des Hs zeichnen
        c.drawForward(size);

        //Zeichne einen H-Tree der Tiefe n-1 an der unteren rechten Ecke.

        drawHTree(hs,n-1);
        c.pop();
    }


}
```

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.

Lösung: