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

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);          4.0          // a)
        System.out.println(b1.f(7d,8L));   13           // b)
        System.out.println(b1.f(10d,17));  11           // c)
        System.out.println(b1.f(5,6L));    12           // 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.

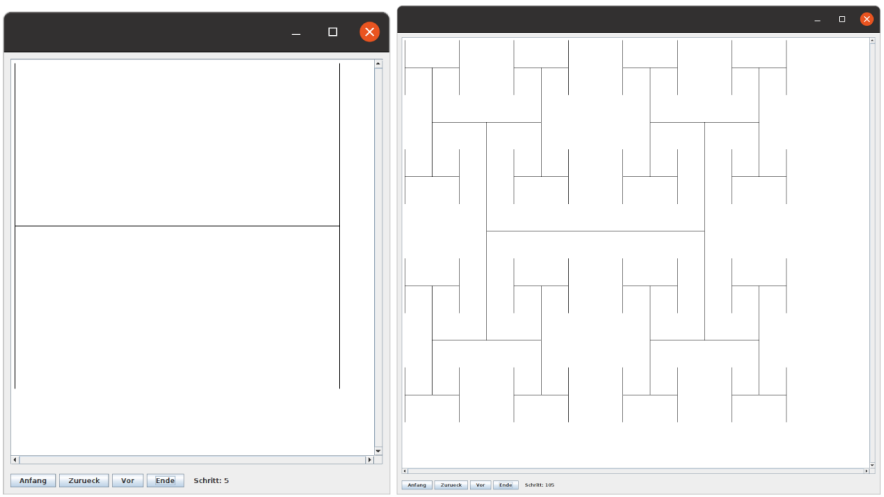

Abbildung 1: Beispiel: Programmausgabe für einen H-Baum der Tiefe 1 (links) und 3 (rechts)

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.[1] 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);
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)
```

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

Lösung:

Listing 1: 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: