Prof. Dr. J. Giesl
S. Dollase, M. Hark, D. Cloerkes

Aufgabe 2 (Rechtecke — VPL): (1 + 3 + 2 + 8 + 3 + 4 + 4 = 25 Punkte)

In dieser Aufgabe soll eine Java-Klasse erstellt werden, mit der sich Rechtecke repräsentieren lassen. Ein solches Rechteck lässt sich mit den Koordinaten x und y für die linke obere Ecke, der Breite *width* und der Höhe *height* beschreiben, wobei x, y, *width* und *height* ganze Zahlen sind. Die Breite und die Höhe eines Rechtecks können nicht negativ sein.
Ihre Implementierung sollte mindestens die folgenden Methoden beinhalten. Sie sollten dabei die Konzepte der Datenkapselung berücksichtigen. Hilfsmethoden müssen als `private` deklariert werden. In dieser Aufgabe dürfen Sie die in der Klasse `Utils` zur Verfügung gestellten Hilfsfunktionen, aber keine Bibliotheksfunktionen verwenden. Sie finden die Klasse im Lernraum.

Die Lösung dieser Aufgabe muss in VPL hochgeladen werden. Beachten Sie dazu die entsprechenden allgemeinen Hinweise auf Seite 1. Der VPL-Editor bietet unter *Ausführen* auch einige einfache Testfälle zum Testen der grundlegenden Funktionaliät Ihrer Implementierung.
Die Testfälle decken aber nicht alle möglichen Fälle ab. Um Ihre Implementierung selbst zu testen, können sie in der Klasse `Rectangle` eine `main`-Methode schreiben und diese unter *Debuggen* mit `run Rectangle` ausführen.

a) Erstellen Sie eine Klasse `Rectangle` mit den Attributen `x`, `y`, `width` und `height`.

b) Erstellen Sie die folgenden öffentlichen Methoden, um Objekte des Typs `Rectangle` erzeugen zu können. Entscheiden Sie dabei selbst, welche Methoden Sie als statisch deklarieren:

```
Rectangle(int xInput, int yInput, int widthInput, int heightInput)
Rectangle(int xInput, int yInput, Integer sidelengthInput)
Rectangle copy(Rectangle toCopy)
```

Beachten Sie dabei folgende Punkte:

- Der Konstruktor `Rectangle` mit vier Argumenten soll ein Rechteck erzeugen, dessen Attribute jeweils die von den entsprechenden Parametern angegebenen Werte haben.
- Der Konstruktor `Rectangle` mit drei Argumenten soll ein Quadrat erzeugen, dessen Höhe und Breite den Wert des Parameters `sidelength` annehmen und dessen übrige Attribute jeweils die von den entsprechenden Parametern angegebenen Werte haben.
- Falls bei den ersten beiden Methoden eines der Argumente einen unzulässigen Wert hat, muss eine geeignete Fehlermeldung ausgegeben und direkt im Anschluss `return` ausgeführt werden. Zur Ausgabe einer Fehlermeldung kann die Methode `Utils.error(String msg)` genutzt werden.
- Die Methode `copy` soll ein Rechteck kopieren.

c) Erstellen Sie die folgenden Selektoren, um die Koordinaten, die Breite und die Höhe eines Rechtecks setzen und auslesen zu können. Entscheiden Sie dabei selbst, welche Methoden Sie als statisch deklarieren:

```
int getX()
void setX(int x)
int getY()
void setY(int y)
int getWidth()
void setWidth(int width)
int getHeight()
void setHeight(int height)
```

Falls ein Attribut auf einen unzulässigen Wert gesetzt werden soll, darf der Wert des Attributes nicht verändert werden. Stattdessen muss eine geeignete Fehlermeldung ausgegeben werden. Hierzu kann die Methode `Utils.error(String msg)` genutzt werden.

d) Erstellen Sie die folgenden Methoden. Entscheiden Sie dabei selbst, welche Methoden Sie als statisch deklarieren und begründen Sie Ihre Entscheidung kurz:

```
Rectangle union(Rectangle... rectangles)
Rectangle intersection(Rectangle... rectangles)
```

Bei den in diesem Aufgabenteil geforderten Methoden muss der Aufrufer (und nicht Sie als Implementierer der Klasse `Rectangle`) sicherstellen, dass die Parameter, mit denen die Methoden aufgerufen werden, nicht den Wert `null` haben bzw. enthalten.

Beachten Sie dabei folgende Punkte:

- Die Methode `union(Rectangle... rectangles)` soll jenes Rechteck zurückgeben, das durch die Vereinigung aller Rechtecke in `rectangles` entsteht. Wenn `rectangles` leer ist, soll `null` zurückgegeben werden.
  Hinweise: Die Vereinigung zweier Rechtecke ist das kleinste Rechteck, das die beiden anderen vollständig abdeckt. Es empfiehlt sich, eine Hilfsmethode zu implementieren, die die Vereinigung *zweier* Rechtecke berechnet.
- Die Methode `intersection(Rectangle... rectangles)` gibt das größte Rechteck zurück, das *vollständig* in *allen* als Argument übergebenen Rechtecken enthalten ist. Wenn `rectangles` leer ist, soll `null` zurückgegeben werden. Falls der Schnitt der Rechtecke leer ist, soll ebenfalls `null` zurückgegeben werden.
  Hinweis: Es empfiehlt sich, eine Hilfsmethode zu implementieren, die den Schnitt *zweier* Rechtecke berechnet.
- Sie finden in der Klasse `Utils` zwei Methoden `min` und `max`, die das Minimum bzw. Maximum von beliebig vielen Werten des Typs `int` berechnen.

Beispiel: Wir betrachten die beiden Rechtecke, die mit den Aufrufen `new Rectangle(1,4,2,3)` und `new Rectangle(2,5,3,3)` erzeugt werden. Das ausgegebene Rechteck beim Aufruf von `union` soll die Werte $1, 5, 4, 4$ haben, beim Aufruf von `intersection` soll ein Rechteck mit den Werten $2, 4, 1, 2$ zurückgegeben werden. Die Werte stehen jeweils in der Reihenfolge $x, y, width, height$. Beim Klick auf *Execution* in VPL werden diese Operationen ausgeführt.

e) Erstellen Sie ebenfalls eine Implementierung für die öffentliche Methode

```
String toString()
```

Entscheiden Sie dabei selbst, ob Sie die Methode als statisch deklarieren. Die Methode `toString()` erstellt eine textuelle Repräsentation des aktuellen Rechtecks. Dies geschieht über die Eckpunkte des Rechtecks, die, beginnend bei der unteren rechten Ecke, gegen den Uhrzeigersinn ausgegeben werden sollen.

Zum Beispiel stellt der String `(6|1),(6|5),(3|5),(3|1)` das Rechteck mit den Koordinaten 3 und 5, der Breite 3 und der Höhe 4 dar.

f) Dokumentieren Sie alle Methoden, die als `public` markiert sind, indem Sie die Implementierung mit Javadoc-Kommentaren ergänzen. Diese Kommentare sollten eine allgemeine Erklärung der Methode sowie weitere Erklärungen jedes Parameters und des `return`-Wertes enthalten. Verwenden Sie innerhalb des Kommentars dafür die Javadoc-Anweisungen `@param` und `@return`.

Benutzen Sie das Programm `javadoc`, um Ihre Javadoc-Kommentare in das HTML-Format zu übersetzen. Überprüfen Sie mit einem Browser, ob das gewünschte Ergebnis generiert wurde. (Falls `javadoc` ihre Abgabe nicht kompiliert, werden **keine** Punkte vergeben.) Bitte drucken Sie die generierten Dateien, der Umwelt zuliebe, *nicht* aus.

g) In dieser Aufgabe geht es um bestimmte Arten von Rechtecken. Wir unterscheiden zwischen den folgenden Arten: Ein Rechteck mit Höhe und Breite 0 ist ein `POINT`, eines mit Höhe und Breite 1 ein `PIXEL`. Ein Rechteck mit Höhe 0 und Breite ≥ 1 ist eine `HLINE` (*horizontale Linie*), der umgekehrte Fall eine `VLINE` (*vertikale Linie*). Ein quadratisches Rechteck mit Seitenlänge ≥ 2 ist ein `SQUARE`. Ein Rechteck mit Höhe 1 und Breite ≥ 2 ist eine `ROW`, der umgekehrte Fall eine `COLUMN`. Erfüllt ein Rechteck keine dieser Bedingungen, fällt es unter `OTHER`.

Um diese verschiedenen Arten von Rechtecken zu modellieren, müssen Sie ein öffentliches `enum` mit dem Namen `RectangleSpecies` erstellen. Schreiben Sie dann eine öffentliche, nicht statische Methode

```
RectangleSpecies determineSpecies()
```

Diese soll das korrekte Objekt vom Typ `RectangleSpecies` für das gegebene `Rectangle` zurückgeben. Ergänzen Sie auch für diese Methode einen geeigneten Javadoc-Kommentar.

Beispiel: Wir betrachten wiederum die beiden Rechtecke, die mit den Aufrufen `new Rectangle(1,4,2,3)` und `new Rectangle(2,5,3,3)` erzeugt werden. Beim Aufruf von `determineSpecies` soll für das erste Rechteck `OTHER` und für das zweite `SQUARE` zurückgegeben werden. Beim Klick auf *Ausführen* in VPL werden diese Operationen ausgeführt.

Aufgabe 4 (Programmanalyse): (7 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. 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 5 (Deck 4): (Codescape)

Lösen Sie die Räume von Deck 4 des Spiels Codescape.
Ihre Lösung für Räume dieses Codescape Decks wird nur dann für die Zulassung gezählt, wenn Sie die Lösung bis Montag, den 25.11.2019, um 12:00 Uhr abschicken.