Objektorientierte Programmierung: Aufgabenblatt 6

Florian Ludewig (185722)

19. Juni 2020

Aufgabe 1

App.java

```java
public class App {
  public static void main(String[] args) throws Exception {
    // contained
    new Rectangle(2.5, 2.5, 4, 4).getLocationRelation(new Rectangle(1, 2.5,
        1, 1));

    // aligned
    new Rectangle(0, 0.5, 3, 2).getLocationRelation(new Rectangle(2, -1, 2,
        2));

    // same
    new Rectangle(2, -2.1, 2, 2).getLocationRelation(new Rectangle(2, -2.1,
        2, 2));

    // disjoint
    new Rectangle(0, 0, 1, 1).getLocationRelation(new Rectangle(-100, 100,
        1, 1));
  }
}
```

Figure.java

```java
abstract class Figure {
  protected double x;
  protected double y;

  Figure() {
    x = 0;
    y = 0;
  }

  Figure(double x, double y) {
    this.x = x;
    this.y = y;
  }

  public double getX() {
    return x;
  }

  public double getY() {
    return y;
  }

  void printXY() {
    System.out.println("X: " + getX() + " Y: " + getY());
  }
}
```

MobileObject.java

```
1   interface MobileObject {
2     void move(double x, double y);
3
4     void increase(double value);
5
6     void decrease(double value);
7   }
```

Rectangle.java

```
1   public class Rectangle extends Figure implements MobileObject {
2     private double height;
3     private double width;
4
5     Rectangle(double h, double w) {
6       super();
7       height = h;
8       width = w;
9     }
10
11    Rectangle(double x, double y, double h, double w) {
12      super(x, y);
13      height = h;
14      width = w;
15    }
16
17    double getHeight() {
18      return height;
19    }
20
21    double getWidth() {
22      return width;
23    }
24
25    void printXY() {
26      super.printXY();
27      System.out.println("Height: " + getHeight() + " Width: " + getWidth());
28    }
29
30    public void move(double x, double y) {
31      this.x += x;
32      this.y += y;
33    }
34
35    public void increase(double value) {
36      width *= value;
37      height *= value;
38    }
39
40    public void decrease(double value) {
41      width *= 1 / value;
42      height *= 1 / value;
43    }
44
45    void getLocationRelation(Rectangle r) {
46      double intersectionY = intersectionYAxis(r);
47      double intersectionX = intersectionXAxis(r);
48
49      if (intersectionY < 0 || intersectionX < 0) {
50        System.out.println("disjoint");
51        return;
52      }
53
54      if (intersectionY == r.getHeight() && intersectionY == getHeight() &&
          intersectionX == r.getWidth()
55          && intersectionX == getWidth()) {
```

```
56        System.out.println("same");
57        return;
58      }
59
60      if (intersectionY == Math.min(r.getHeight(), getHeight()) &&
            intersectionX == Math.min(r.getWidth(), getWidth())) {
61        System.out.println("contained");
62        return;
63      }
64
65      if ((intersectionX == 0 && intersectionY != 0) || (intersectionX != 0 &&
            intersectionY == 0)) {
66        System.out.println("aligned");
67        return;
68      }
69
70      System.out.println("something else");
71    }
72
73    private double intersectionYAxis(Rectangle r) {
74      return getIntersection(y - getHeight() / 2, y + getHeight() / 2, r.y - r
            .getHeight() / 2, r.y + r.getHeight() / 2);
75    }
76
77    private double intersectionXAxis(Rectangle r) {
78      return getIntersection(x - getWidth() / 2, x + getWidth() / 2, r.x - r.
            getWidth() / 2, r.x + r.getWidth() / 2);
79    }
80
81    private double getIntersection(double min1, double max1, double min2,
        double max2) {
82      if (max2 > max1 && min1 > min2)
83        return max1 - min1;
84      if (max1 > max2 && min2 > min1)
85        return max2 - min2;
86      return max1 > max2 ? max2 - min1 : max1 - min2;
87    }
88  }
```

Aufgabe 2

```
Roman.java
```

```
1  public class Roman {
2    private String romanString;
3    private int romanValue;
4
5    Roman(String roman) {
6      this.romanString = roman;
7      romanValue = romanToInt(roman);
8    }
9
10   Roman(int value) {
11     romanValue = value;
12     romanString = intToRoman(value);
13   }
14
15   public Roman add(Roman r) {
16     return new Roman(getValue() + r.getValue());
17   }
18
19   public Roman substract(Roman r) {
20     int result = getValue() - r.getValue();
21     if (result <= 0)
22       throw new Error("negative numbers or zero no allowed");
```

```java
23        return new Roman(result);
24      }
25
26      public Roman multiply(Roman r) {
27        return new Roman(getValue() * r.getValue());
28      }
29
30      public Roman divide(Roman r) {
31        return new Roman(getValue() / r.getValue());
32      }
33
34      public String toString() {
35        return romanString;
36      }
37
38      public int hashCode() {
39        return getValue();
40      }
41
42      public int getValue() {
43        return romanValue;
44      }
45
46      public boolean equals(Roman r) {
47        return r.hashCode() == hashCode();
48      }
49
50      private String intToRoman(int value) {
51        String roman = "";
52        String[] chars = new String[] { "I", "IV", "V", "IX", "X", "XL", "L", "
             XC", "C", "CD", "D", "CM", "M" };
53        int[] values = new int[] { 1, 4, 5, 9, 10, 40, 50, 90, 100, 400, 500,
             900, 1000 };
54
55        int counter = 0;
56        for (int i = values.length - 1; i >= 0; i--) {
57          counter = value / values[i];
58          value = value % values[i];
59          for (int j = 0; j < counter; j++)
60            roman += chars[i];
61        }
62        return roman;
63      }
64
65      private int romanToInt(String roman) {
66        int value = 0;
67        String[] binaries = new String[] { "IV", "IX", "XL", "XC", "CD", "CM" };
68
69        while (roman.length() > 0) {
70          if (roman.length() > 1 && stringIncludes(roman.substring(0, 2),
               binaries)) {
71            value += romanCharToValue(roman.substring(0, 2));
72            roman = roman.substring(2, roman.length());
73          } else {
74            value += romanCharToValue(roman.substring(0, 1));
75            roman = roman.substring(1, roman.length());
76          }
77        }
78
79        return value;
80      }
81
82      private int romanCharToValue(String romanChar) {
83        String[] chars = new String[] { "I", "IV", "V", "IX", "X", "XL", "L", "
             XC", "C", "CD", "D", "CM", "M" };
```

```java
84       int[] values = new int[] { 1, 4, 5, 9, 10, 40, 50, 90, 100, 400, 500,
            900, 1000 };
85       for (int i = 0; i < chars.length; i++) {
86         if (romanChar.equals(chars[i]))
87           return values[i];
88       }
89       return 0;
90     }
91
92     private boolean stringIncludes(String search, String[] strings) {
93       for (String s : strings) {
94         if (s.equals(search))
95           return true;
96       }
97       return false;
98     }
99   }
```