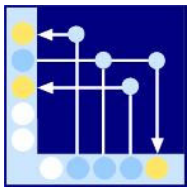
Hochschule Aalen
Fakultät Elektronik und Informatik
Studiengang Informatik



Programmieren 2

Vorlesung im Wintersemester 2014/2015
Prof. Dr. habil. Christian Heinlein

4. Übungsblatt (4. November 2014)

Aufgabe 4: Klassen

Teilaufgabe 4.a)

Implementieren Sie eine Java-Klasse `Point` zur Repräsentation zweidimensionaler Punkte, die wie folgt verwendet werden kann:

```
// Punkt p1 mit x-Koordinate 1.5 und y-Koordinate -3.7 erzeugen.
Point p1 = new Point(1.5, -3.7);

// Punkt p2 mit x- und y-Koordinate 0 erzeugen.
Point p2 = new Point();

// Koordinaten von p1 abfragen und ausgeben.
System.out.println("x-Koordinate: " + p1.getX());
System.out.println("y-Koordinate: " + p1.getY());

// Punkt p1 in x-Richtung um 2.5 und in y-Richtung um 0.7 verschieben.
p1.move(2.5, 0.7);

// Punkt p1 ausgeben.
p1.print();             // Ausgabe muss lauten: (4.0, -3.0)
```

Teilaufgabe 4.b)

Implementieren Sie eine weitere Klasse `Line` zur Repräsentation gerader Linien, die wie folgt verwendet werden kann:

```
// Linie mit Anfangspunkt p1 und Endpunkt p2 erzeugen.
Line ln = new Line(p1, p2);

// Anfangs- und Endpunkt von ln abfragen und ausgeben.
ln.getBegin().print();  // Ausgabe: (4.0, -3.0)
ln.getEnd().print();    // Ausgabe: (0.0, 0.0)

// Länge von ln berechnen und ausgeben.
System.out.println("Länge: " + ln.length());
                        // Ausgabe: 5.0
```

Teilaufgabe 4.c)

Schreiben Sie eine dritte Klasse `Test` mit einer Hauptmethode `main` zum Testen der beiden anderen Klassen!

Hinweise

- Die Klassen `Point` und `Line` bestehen jeweils aus:
  - privaten Objektvariablen (z. B. `x` und `y` mit Typ `double` zur Speicherung der Koordinaten eines Punkts),
  - einem oder mehreren öffentlichen Konstruktoren zur Initialisierung der Objektvariablen
  - und öffentlichen Objektmethoden, die die Objektvariablen verwenden und eventuell verändern.
- `Math.sqrt(z)` berechnet die Quadratwurzel des `double`-Werts `z`.
- Sie können entweder jede Klasse in eine eigene Datei schreiben und beim Übersetzen alle Dateien angeben, zum Beispiel:

  ```
  javac Point.java Line.java Test.java
  java Test
  ```

  Alternativ können Sie alle Klassen in eine einzige Datei mit beliebigem Namen schreiben und nur diese übersetzen, zum Beispiel:

  ```
  javac aufgabe4.java
  java Test
  ```

  Zum Ausführen des Programms muss in jedem Fall der Name der Klasse angegeben werden, die die Hauptmethode `main` enthält.

Lösung

```
// Punkt im zweidimensionalen Raum.
class Point {
    // x- und y-Koordinate.
    private double x, y;

    // Neuer Punkt mit Koordinaten x und y.
    public Point (double x, double y) {
        this.x = x;
        this.y = y;
    }

    // Neuer Punkt mit Koordinaten 0 und 0.
    public Point () {
        this(0, 0);
    }

    // Koordinaten abfragen.
    public double getX () { return x; }
    public double getY () { return y; }
```

```
    // Punkt in x-Richtung um dx und in y-Richtung um dy verschieben.
    public void move (double dx, double dy) {
        x += dx;
        y += dy;
    }

    // Punkt in der Form (x, y) ausgeben.
    public void print () {
        System.out.println("(" + x + ", " + y + ")");
    }
}

// Linie im zweidimensionalen Raum.
class Line {
    // Anfangs- und Endpunkt.
    private Point begin, end;

    // Neue Linie mit Anfangspunkt begin und Endpunkt end.
    public Line (Point begin, Point end) {
        this.begin = begin;
        this.end = end;
    }

    // Anfangs- und Endpunkt abfragen.
    public Point getBegin () { return begin; }
    public Point getEnd () { return end; }

    // Länge der Linie berechnen und zurückliefern.
    public double length () {
        double dx = begin.getX() - end.getX();
        double dy = begin.getY() - end.getY();
        return Math.sqrt(dx * dx + dy * dy);
    }
}

// Test der Klassen Point und Line.
class Test {
    public static void main (String [] args) {
        // Punkt p1 mit x-Koordinate 1.5 und y-Koordinate -3.7 erzeugen.
        Point p1 = new Point(1.5, -3.7);

        // Punkt p2 mit x- und y-Koordinate 0 erzeugen.
        Point p2 = new Point();

        // Koordinaten von p1 abfragen und ausgeben.
        System.out.println("x-Koordinate: " + p1.getX());
        System.out.println("y-Koordinate: " + p1.getY());

        // Punkt p1 in x-Richtung um 2.5 und in y-Richtung um 1.2 verschieben.
        p1.move(2.5, 0.7);

        // Punkt p1 ausgeben.
        p1.print();             // Ausgabe muss lauten: (4.0, -3.0)
```

```
        // Linie mit Anfangspunkt p1 und Endpunkt p2 erzeugen.
        Line ln = new Line(p1, p2);

        // Anfangs- und Endpunkt von ln abfragen und ausgeben.
        ln.getBegin().print();  // Ausgabe: (4.0, -3.0)
        ln.getEnd().print();    // Ausgabe: (0.0, 0.0)

        // Länge von ln berechnen und ausgeben.
        System.out.println("Länge: " + ln.length());
                                // Ausgabe: 5.0
    }
}
```