Grundlagen von Java

Aufgabe 1: Verständnisfragen

Welche der folgenden Aussagen sind korrekt? Kreuzen Sie alle zutreffenden an. (Können Sie Ihre Antworten auch begründen?)

- ☐ a) Konstruktormethoden dienen der Erzeugung von Objekten.
- ☐ b) Der Programmierer muss immer einen Konstruktor erstellen.
- ☐ c) Jede Klasse besitzt mindestens einen Konstruktor.
- ☐ d) Nur Konstruktormethoden können überladen werden.
- ☐ e) Als private deklarierte Attribute können nur innerhalb der eigenen Klasse gelesen werden.
- ☐ f) In als public deklarierten Methoden darf nicht auf die Attribute zugriffen werden.

Aufgabe 2: Fehler

Klaus Schussel hat im folgenden Programm eine Unmenge an Fehlern eingebaut. Können Sie Ihm helfen, die Fehler zu finden und zu korrigieren?

```
public class A {
     public String msg;
     private int len;

     private A(String msg) {
          len = this.msg.length();
          msg = msg;
     }

     public int f(int i) {
          return i + len;
     }
     public String f(int i) {
          return i + msg;
     }

     private void g(int i) {
          int value;
          if (i >=0)
               value = i;
          len = value;
          return result;
     }
}
```

```
class B {
     private static void main(String[] args) {
          A a = new A();
          g(17);
          String msg = a.f(0);
     }
}
```

Aufgabe 3: Java Klassendefinition

Für Spiele und andere grafische Anwendungen müssen wir Objekte auf der Karte positionieren und ggf. bewegen können. Dazu ist u.a. deren Position erforderlich.

Teil 1. Erstellen Sie eine Klasse **`Point`**, welche die Position bzw. Koordinaten im zweidimensionalen Raum realisiert.

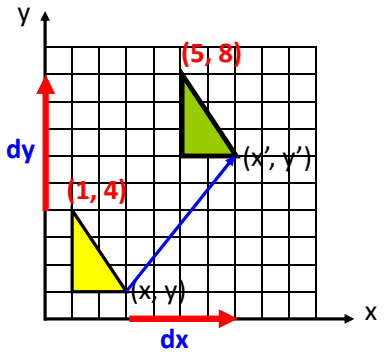
Als **Attribute** sind die jeweiligen **`x`**- bzw. **`y`**-Koordinaten zu speichern, Dabei sollen die internen Werte der Koordinaten von außen nicht direkt zugreifbar sein.

Die Klasse Point soll ferner die folgenden **Methoden** bereitstellen:

- **`Konstruktormethoden`** zum Setzen der Koordinaten. Dabei soll auch ein Default-Konstruktor mit dem Koordinatenursprung (0,0) angeboten werden.
- **`get-Methoden`** zum Lesen der jeweiligen x- und y-Koordinaten.
- **`public String toString()`** soll eine textuelle Repräsentation des Objektes zurückliefern.

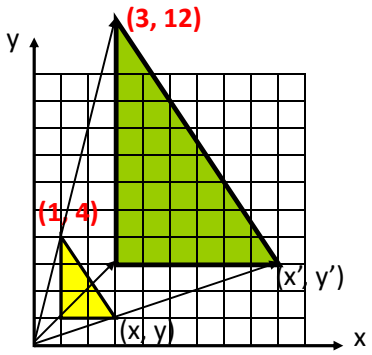
- **`Point move(Point shift)`** soll die Koordinaten entsprechend des anderen Punktes (= Verschiebevektors) shift verschieben: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x + dx \\ y + dy \end{pmatrix}$

  Anmerkung: Falls jemand mit der verschoben Position weiterarbeiten will, geben wir das geänderte Objekt zurück.



- **`Point moveTo(Point newPos)`** soll die Koordinaten zum anderen Punkt newPos verschieben: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} newPosX \\ newPosY \end{pmatrix}$

- **`Point scale(double factor)`** skaliert die Koordinaten:
  $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \alpha \bullet x \\ \alpha \bullet y \end{pmatrix}$



- **`double distance()`** liefert den Abstand $\sqrt{x^2 + y^2}$ des Punktes vom Koordinatenursprung (0,0).
- **`double distance(Point other)`** liefert den Abstand der beiden Punkte voneinander.

- **`boolean equals(Point other)`** zum Vergleich zweier Objekte.

Teil 2: Testen Sie Ihr Programm.

Was würde beispielsweise der folgende Test ergeben?

```
Point p = new Point(1, 4);
p.scale(2);
p.move( new Point(2, -5) );
Point q = new Point();
System.out.println("p = " + p);
System.out.println("q = " + q);
System.out.println("Distance = " + p.distance(q) );
System.out.println("p.equals(q) = " + p.equals(q) );
```

Zusatz-Aufgabe 4: Schiffe versenken

Falls Sie noch Zeit und Lust haben, können Sie Ihr SchiffeVersenken1D-Programm von Übung 1 um eine Klasse **`Ship`** zur Repräsentation von Schiffen (oder ähnlichen Spielobjekten) erweitern. Diese besitzt intern (als Attribute)

- einen Namen
- ihre momentane Position
- die Größe (d.h. Ausdehnung) in x- bzw. y-Richtung

sowie die folgenden Methoden:

- ein geeigneter **`Konstruktor`**
- **`get-Methoden`** (zum Lesen des Namens, der Position und Größe)
- eine **`toString()-`**Methode
- eine Testfunktion **`boolean overlapps(Ship other)`** zur Überprüfung, ob sich zwei Schiffe gegenseitig rammen würden, d.h. sich deren Positionen partiell überschneiden.

Zusatz-Aufgabe 5 (für Java-Experten): Java-Rätsel – Strings/Konstruktoren

a) Rätsel (Strings): Was gibt das folgende Programm aus – und warum?

```
public class Lachen {
	public static void main(String[] args) {
		System.out.print("H" + "a");
		System.out.print('H' + 'a');
	}
}
```

b) Rätsel (Strings): Was gibt das folgende Programm aus – und warum?

```
public class Abc {
	public static void main(String[] args) {
		String letters = "ABC";
		char [] numbers = {'1', '2', '3'};
		System.out.println(letters + " easy as " + numbers);
	}
}
```

c) Rätsel (Konstruktor): Die folgende Klasse besitzt zwei überladene Konstruktoren. Die main-Methode ruft einen Konstruktor auf – aber welchen?
Was gibt das Programm aus? Ist es überhaupt ein korrektes Java-Programm?

```
public class Confusing {
	private Confusing(Object o) {
		System.out.println("Object");
	}

	private Confusing(double[] dArray) {
		System.out.println("double array");
	}

	public static void main(String[] args) {
		new Confusing(null);
	}
}
```