Übung07: Klassen im Sinne von OOP

Abgabetermin: 10.05.19, 15:30 Uhr

Name: _____ Matrikelnummer: _____

Aufgabe	**schriftlich abzugeben**	**elektronisch abzugeben**	**gelöst**
Aufgabe19	Java Programm, Testausgaben	Java Programm	☐
Aufgabe20	Java Programm, Bildschirmausgaben	Java Programm	☐

Achtung!

Bitte auf diesem Deckblatt:

- **Name** und **Matrikelnummer** ausfüllen,
- gelöste Aufgaben **ankreuzen**

und dann der schriftlichen Abgabe als erste Seite anheften.

Aufgabe19: Berechnungen mit komplexen Zahlen (10 Punkte)

Gegeben ist eine einfache Klasse `Complex` für Berechnungen mit komplexen Zahlen:

```
class Complex {
  final float real;
  final float imag;
}
```

Erweitern Sie die Klasse um folgende objektorientierte Inhalte:

1) Definieren Sie einen Konstruktor `Complex(float real, float imag)`, der ein neues Objekt erzeugt und dabei die beiden gegebenen Parameter den Objektfeldern zuweist.

2) Definieren Sie einen parameterlosen Default-Konstruktor, der ein neues Objekt erzeugt, wobei die beiden Objektfelder `real` und `imag` mit `0f` initialisiert werden. Duplizieren Sie hier nicht den Code vom oberen Konstruktor, sonden rufen Sie diesen mit den entsprechenden Werten auf.

3) Implementieren Sie eine Methode `Complex add(Complex other)`, die ein neues Objekt erzeugt und zurückliefert, indem Sie jeweils Real- und Imaginärwert des Empfängers (`this`) und des übergebenen Objekts (`other`) zusammenzählen.

4) Überschreiben Sie die Methode `public String toString()` der Basisklasse `Object` und liefern Sie eine String-Representation für komplexe Zahlen in dem Format `"%f + %fi"` zurück. Beispiel: `5.0 + 3.0i`. Achten Sie auf Sonderfälle, wenn z.B. Real- oder Imaginärteil null, oder der Imaginärteil negativ ist.

5) Überschreiben Sie die Methode `public boolean equals(Object other)` der Basisklasse `Object` und prüfen Sie, ob das übergebene Objekt eine komplexe Zahl darstellt und ob die Real- und Imaginärwerte dieser Zahl den Werten des Empfängers gleichen. Führen Sie hierbei eine Typprüfung und Typumwandlung zur Laufzeit durch und vergleichen Sie dann die Werte der Objekte.

Testen Sie ihre Klasse, indem Sie darin eine main-Methode definieren, verschiedene Objekte für komplexe Zahlen erzeugen und die oben definierten Inhalte ausprobieren.

Abzugeben sind:

- Java Programm
- Testausgaben

Aufgabe20: Klassenhierarchie für Shapes (14 Punkte)

In der letzten Übung wurde ein Programm entwickelt, mit dem sich einfache Shapes zeichnen lassen, wobei die Shape-Klassen als reine Datenstrukturen verwendet wurden. In dieser Aufgabe soll ein erweitertes Programm entwickelt werden, in dem die Shape-Typen in einer gemeinsamen Klassenhierarchie abgebildet werden. Gegeben sind folgende Klassen `Point` und `Shape`:

```
final class Point {
  int x, y;

  Point(int x, int y) {
    this.x = x; this.y = y;
  }

  void move(int dx, int dy) {
    x += dx; y += dy;
  }
}

abstract class Shape {
  Point position;

  Shape(Point position) {
    this.position = position;
  }

  void move(int dx, int dy) {
    position.move(dx, dy);
  }

  abstract void draw();
  abstract boolean hit(Point point);
}
```

Klasse `Shape` ist hierbei die abstrakte Basisklasse. Sie implementiert bereits eine allgemeine Positionslogik, die für alle Shapes gültig ist, und definiert zwei abstrakte Methoden `draw` und `hit`, die von den konkreten Subklassen implementiert werden müssen. Letzere liefert `true`, falls sich der gegebene Punkt im Empfängerobjekt befindet.

Erweitern Sie das Programm um folgende objektorientierte Inhalte:

1) Erstellen Sie zwei Klassen `Circle` (`final`) und `Rectangle`, und lassen Sie diese von `Shape` erben. `Circle` soll neben der Position einen Radius, und `Rectangle` eine Breite und Höhe besitzen. Definieren Sie dafür in den beiden Klassen jeweilige Felder, sowie einen Konstruktor mit entsprechenden Parametern. Um die Position zu setzen, welche in `Shape` definiert ist, muss der Superkonstruktor von `Shape` aufgerufen werden.

2) Implementieren Sie die Methoden `draw` und `hit` konkret für `Circle` und `Rectangle`. Verwenden Sie für `draw` die bereits bekannte `Window` Klasse.

3) Implementieren Sie eine `final` Klasse `Square` als Spezialfall von `Rectangle`, wo die Breite und Höhe gleich ist. Dabei soll `Square` lediglich ein Feld für die Seitenlänge besitzen und einen entsprechenden Konstruktor definieren, der den Superkonstruktor von `Rectangle` entsprechend aufruft.

4) Implementieren Sie eine Klasse `Group`, die mehrere `Shapes` zusammenfasst. `Group` soll dabei ein `Shape`-Array als Feld, sowie einen entsprechenden Konstruktor, wo ein solches Array übergeben wird, besitzen.

Achtung: Die übergebenen `Shape`-Objekte sollen relativ zur Position des `Group`-Objekts stehen, d.h. im Konstruktor von `Group` müssen die übergebenen `Shapes` um die Position des `Group`-Objekts verschoben werden. Stellen Sie sicher, dass dies im Konstruktor passiert.

5) Implementieren Sie die Methoden `draw` und `hit` konkret für `Group`. Dabei soll `Group` den `draw` und `hit` Aufruf über dynamische Bindung an die konkreten `Shape`-Objekte im Array weiter delegieren.

6) Überschreiben Sie in der Klasse `Group` die Methode `move`, die bereits in der Klasse `Shape` definiert wurde, damit nicht nur die Position des `Group`-Objekts, sondern auch die der `Shape`-Objekte im Array verschoben wird, wenn `move` auf einem `Group`-Objekt aufgerufen wird.

Testen Sie ihr Programm, indem Sie in einer main-Methode unterschiedliche `Shape`-Objekte, sowie Gruppen von `Shapes` erzeugen, diese zeichnen, verschieben, usw. Sie können auch versuchen, das Programm der letzten Übung anzupassen, um Shapes aus einer Datei einzulesen.

Im Moodle befindet sich eine Datei `HitTheSnowMan.java` aus dem letzten Wintersemester, mit der Sie Ihr Programm zusätzlich testen können. Kopieren Sie dafür die Datei in ihr Projektverzeichnis und führen Sie die darin enthaltene main-Methode aus. Die Klasse `HitTheSnowMan` implementiert ein kleines Spiel, in dem man versuchen muss, einen Schneemann mit Schnebällen zu treffen.

Abzugeben sind:

- Java Programm
- Bildschirmausgaben