HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Heinrich-Heine-Universität Düsseldorf
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik, Abteilung Betriebssysteme
Prof. Dr. Michael Schöttner
Kevin Beineke
Ludmila Himmelspach
Stefan Nothaas

7. Übungsblatt „Programmierung“ im Wintersemester 2016/17

Abgabe bis 05. Dezember 2016, 07:59 Uhr über das Abgabesystem

Aufgabe 1: Geometrie - Eigene Klassen — 20 Punkte

In dieser Aufgabe sollen Sie eigene Klassen in einem Package `geometry` erstellen. Folgende Klassen mit den angegebenen Methoden sind in separaten `.java`-Dateien zu implementieren:

Point: Ein Punkt mit den Koordinaten x und y (je float)

- Konstruktor: Bekommt als Parameter die float-Koordinaten x und y
- Getter-Methoden für die Koordinaten
- `boolean equals(Object o)`: Überprüft, ob zwei Punkte identisch sind (Koordinaten identisch)
- `String toString()`: Erzeugt eine lesbare Darstellung

Line: Eine Line, bestehend aus zwei Punkten

- Konstruktor: Bekommt zwei Instanzen von `Point`
- Getter-Methoden für die Punkte
- `float length()`: Berechnt die Länge der Linie.
- `boolean equals(Object o)`: Überprüft, ob zwei Linien identisch sind (Koordinaten identisch)
- `String toString()`: Erzeugt eine lesbare Darstellung

Triangle: Ein Dreieck, bestehend aus drei Punkten

- Konstruktor: Bekommt drei Instanzen von `Point`
- Getter-Methoden für die Punkte
- `Line[] getLines()`: Erzeugt neue Instanzen der Klasse Linie, welche der Verbindungslinien der Punkte des Dreiecks entsprechen
- `boolean equals(Object o)`: Überprüft, ob zwei Dreiecke identisch sind (Koordinaten identisch)
- `String toString()`: Erzeugt eine lesbare Darstellung

Hinweise:

- Arbeiten Sie die Klassen in angegebener Reihenfolge ab. Jede nachfolgende benötigt die vorhergehende Klasse.
- Achten Sie darauf Ihren Methoden und Variablen sinnvolle Sichtbarkeiten (public, protected, private) zu geben.

- Bei den Methoden `boolean equals(Object o)` und `String toString()` müssen Sie "@Override" voranstellen:

```
@Override
public boolean equals(Object o) {
  ...
}
```

- Da die Methode `equals` von Java vorgegeben den Parameter `Object o` bekommt, müssen Sie den Typ in der jeweiligen Klasse passend durch einen Typecast umwandeln:

```
@Override
public boolean equals(Object o) {
  // Hier als Beispiel für die Klasse Point, für alle anderen analog
  Point p = (Point) o;
  ...
}
```

Aufgabe 2: Unit-Testing — 20 Punkte

In dieser Aufgabe sollen die von Ihnen implementierten Klassen (Point, Line, Triangle) aus der vorherigen Aufgabe mithilfe von Unit-Tests (JUnit) getestet werden. Falls Sie die hierfür notwendige Aufgabe nicht gelöst haben, so nutzen Sie die von uns bereitgestellten kompilierten .class-Dateien. Beachten Sie, dass sich diese in einem Verzeichnis mit dem Namen `geometry` befinden müssen. Schreiben Sie nun für jede Ihrer Klassen separate JUnit-Tests mit den nachfolgenden Methoden, die die jeweiligen Methoden der Klasse auf mögliche Eingaben testen.

- **PointTest**: `creation` (Erzeugung von Instanzen und Gültigkeit deren Werte), `equals` (Gleichheit und Ungleichheit)
- **LineTest**: `creation` (Erzeugung von Instanzen und Gültigkeit deren Werte), `equals` (Gleichheit und Ungleichheit), `length` (Korrektheit der Länge)
- **TriangleTest**: `creation` (Erzeugung von Instanzen und Gültigkeit deren Werte), `equals` (Gleichheit und Ungleichheit), `getLines` (Korrektheit der Kanten des Dreiecks)

Hinweise: Bei Verwendung der vorgegebenen `.class`-Dateien haben die Getter folgende Signatur(en):

- **Point**: `float getX()`, `float getY()`
- **Line**: `Point getPointA()`, `Point getPointB()`
- **Triangle**: `Point getPointA()`, `Point getPointB()`, `Point getPointC()`

Aufgabe 3: Minesweeper - Teil 3 — 20 Punkte

Erweitern Sie die Lösung von "Minesweeper, Teil 2", so dass ein interaktives Spiel entsteht. Dazu sollten die Größe des Spielfeldes und die Anzahl an Minen als Argumente übergeben werden und zu Beginn ein komplett verdecktes Feld angezeigt werden. Anschließend soll es dem Spieler ermöglicht werden, per Mausklick auf ein Feld, dieses aufzudecken. Befindet sich darunter eine Mine, soll das Spiel beendet werden oder dem Spieler angezeigt werden, dass er verloren hat. Befindet sich keine Mine darunter, soll angegeben werden, wie viele Minen in der Nachbarschaft sind, indem die Zahl in das Feld geschrieben wird. Wenn alle Felder ohne Minen aufgedeckt wurden, ohne zwischendurch eine Mine aufgedeckt zu haben, soll dem Spieler mitgeteilt werden, dass er gewonnen hat. Spielen Sie dazu auch die Audiodatei cheering.wav ab.

Hinweise:

- Nutzen Sie die Methoden `mousePressed`, `mouseX` und `mouseY` aus der Klasse `StdDraw` für die Erkennung und Lokalisierung eines Mausklick.
- Legen Sie ein zweites zweidimensionales Array an, in dem Sie sich merken, welche Felder bereits aufgedeckt wurden.