Übungen zur Einführung in die Programmiersprache Java

Universität Regensburg
NWF II - Physik
Dominik Köppl

Wintersemester 2011/12
Blatt 9

---

33 Aufgabe zum Abgeben

Beantworten Sie bitte folgende Aufgaben auf einem Papierzettel und geben Sie diesen in der nächsten Stunde ab.

1. Aufgabe
   (a) Welche Möglichkeiten können Sie in der Programmiersprache Java nutzen, um Attribute einer Klasse von außen zugänglich zu machen?
   (b) Welche dieser Möglichkeiten sollten Sie bevorzugen? Begründen Sie Ihre Entscheidung!
2. Aufgabe
   (a) In Bezug auf die Attribute und Methoden einer Klasse spricht man in der objektorientierten Programmierung von Kapselung. Was verstehen Sie darunter?
   (b) Erläutern Sie den Begriff im Vergleich mit der prozeduralen Programmierung [1]!
3. Aufgabe
   (a) Definieren Sie den Begriff "Vererbung" aus der objektorientierten Programmierung.
   (b) Funktioniert "Vererbung" auch in der prozeduralen Programmierung?
4. Aufgabe
   (a) Beschreiben Sie den Ausdruck "Polymorphie" anhand eines Beispiels!
   (b) Gibt es in Java eine Möglichkeit, den Gebrauch der Polymorphie bei bestimmten Methoden/Klassen zu deaktivieren?
   (c) Worin unterscheidet sich die Polymorphie mit dem sog. "Upcasting" ?
5. Aufgabe
   Erläutern Sie die Funktionsweise von "Generics" am Interface `Comparable<T>`!

Auf diese Aufgabe gibt es 2 Punkte

[1] Damit sind Programmiersprachen wie C, Pascal, etc. gemeint - bzw. der Code-Stil, den wir bis zum Kapitel "OOP" studiert haben.

34 TicTacToe I

Das Spielfeld eines *TicTacToe* Spiels besteht aus 3x3 Feldern, die drei verschiedene Zustände haben (unbesetzt oder besetzt von Spieler 1 bzw. 2).

(a) Schreiben Sie eine `Enum Status`, die alle Zustände klassifiziert.

(b) Legen Sie in Ihrer Klasse `Spielfeld` ein Array aus `Status`-Elementen an. Das Spielfeld hat die Methoden `set(int x, int y, Status status)` und `Status get(int x, int y)` zum Überprüfen eines Feldes.

(c) Zudem kann mit `int gameOver()` überprüft werden, ob das Spiel bereits zu Ende ist. Ist das Spiel noch am Laufen wird -1 zurückgegeben. Wurde das Spiel gewonnen, so wird 1 bzw. 2 respektive Spieler 1 bzw. Spieler 2 zurückgegeben. Ansonsten muss ein unentschieden vorliegen, s.d. die Methode den Wert 0 zurückgibt. Das Spiel ist gewonnen, wenn einer der beiden Spieler senkrecht, horizontal oder in einer Diagonalen nebeneinander die Felder belegen konnte. Wurden alle leeren Felder belegt, ist das Spiel auf jeden Fall beendet.

(d) Das Spielbrett kann mit `toString()` ausgegeben werden. Hierbei soll die `toString`-Methode des Enums `Status` benutzt werden, die von Ihnen noch überschrieben werden muss.

(e) Schließlich darf in Ihrer `main`-Routine abwechselnd jeder Spieler ein Feld besetzen.

35 Geraden

Gegeben sei das Interface

```
interface Point<F extends Field<F>> {
        public F getX();
        public F getY();
}
```

Hierbei verwenden wir das Interface `Field`, das wir in einer vorigen Aufgabe bereits geschrieben haben.

(a) Implementieren Sie das Interface `Point` für eine Punktdarstellung. Dazu benötigen Sie zwei passende Attribute. Zusätzlich besitzt Ihre Generics-Klasse einen vollständigen Konvertierungskonstruktor.

(b) Schreiben Sie eine Klasse `Gerade<F extends Field<F>>`, die die Abbildung $g : F \to F, x \mapsto a+bx$ repräsentiert. Zur Berechnung von $a, b \in F$ benötigt der Konstruktor von `Gerade<F>` zwei Punkte $p_i = (x_i, y_i)$. Die Gerade selbst speichert aber nur Aufpunkt und Steigung. Die Steigung wird mit der Formel $b := \frac{y_1-y_2}{x_1-x_2}$ ermittelt. Der Aufpunkt lässt sich dann durch Einsetzen eines Punktes in die Geradengleichung berechnen.

(c) Die Gerade soll mit einer Methode `F at(F x)` den Wert $g(x)$ zurückgeben können.

(d) Testen Sie die Gerade an einer passenden Implementierung! Verwenden Sie dazu eine Implementierung von `Field`.

Auf diese Aufgabe gibt es offiziell einen Punkt. Für die Bearbeitung wird aber ein zusätzlicher Bonuspunkt vergeben.