Grundlagen der Programmkonstruktion
Übungsblatt 3

Organisatorisches

Ablauf

Lösen Sie die Aufgaben in Ihrer Kleingruppe. In der dritten Übungsrunde vom 25. April bis 7. Mai sollen Sie – jedes einzelne Mitglied Ihrer Kleingruppe – in der Lage sein, Lösungen der hier gestellten Aufgaben zu präsentieren, wobei die Lösungswege im Vordergrund stehen, weniger die korrekten Ergebnisse.

Fragen

Bei Fragen wenden Sie sich bitte in den Sprechstunden an die Tutoren (Termine siehe `http://www.complang.tuwien.ac.at/franz/pk#tutoren`). Sie können natürlich auch in die Sprechstunden von Gmeiner und Puntigam kommen (jeweils montags von 10 bis 11 Uhr). Sollte in der Angabe etwas unklar sein, sehen Sie bitte zunächst auf der Homepage der Lehrveranstaltung nach, ob dort bereits ergänzende Anmerkungen stehen. Wichtige Ankündigungen zum Übungsteil werden auch in der Vorlesung gemacht.

Aufgabe 1: Klassen und Objekte (3)

Erweitern Sie die Beispielprogramme aus der Vorlesung vom 29. März 2012:

Aufgabe 1.1: Erstellen eines Interfaces (0.5)

Erstellen Sie ein Interface `Spiegelbar` mit zwei Methoden:

- Eine Methode soll das Objekt am Ursprung spiegeln (Punktspiegelung).
- Eine Methode soll das Objekt an einer Achse spiegeln. Übergeben Sie dieser Methode einen Parameter, mit dem Sie auswählen, ob das Objekt an der x-Achse oder der y-Achse gespiegelt werden soll. Der Typ und die möglichen Werte des Parameters bleiben Ihnen überlassen, Sie sollen aber Ihre Designentscheidung begründen können.

Aufgabe 1.2: Implementieren des Interfaces (1)

Ändern Sie die Klassen `Punkt` und `Scheibe` so ab, dass diese das Interface `Spiegelbar` implementieren.

Beispiele

Ein Punkt mit den Koordinaten $(2, 4)$, der am Ursprung gespiegelt wird (Punktspiegelung) hat nach der Spiegelung die Koordinaten $(-2, -4)$.

Eine Scheibe, die an der x-Achse gespiegelt wird, hat nach der Spiegelung den selben Radius und Abstand vom Ursprung, der Winkel ändert sich allerdings auf `2 * Math.PI - winkel` (2π im Bogenmaß entsprechen 360 Grad).

Aufgabe 1.3: Erstellen einer Klasse `Linie` (1.5)

Erstellen Sie eine Klasse `Linie`. Jede Linie ist durch ihre beiden Endpunkte (Objekte vom Typ `Punkt`) definiert, die dem Konstruktor übergeben werden.

Die Klasse `Linie` soll die Interfaces `Verschiebbar` und `Spiegelbar` implementieren. Gehen Sie davon aus, dass auch `Punkt` diese Interfaces implementiert.

Beachten Sie, dass Objektvariablen `private` sein müssen, und versuchen Sie, Getter- und Setter-Methoden (also Methoden, die nur Werte von Objektvariablen zurückgeben bzw. neu setzen) zu vermeiden.

Aufgabe 2: Bedingte Anweisungen, Schleifen (2)

Aufgabe 2.1: `if` in `switch` umwandeln (0.5)

Wandeln Sie folgende geschachtelte `if`-Anweisungen in eine einzige `switch`-Anweisung mit gleicher Semantik um und vermeiden Sie dabei *Fall-Through*:

```
int i = /* hier sollte ein Wert stehen */;

if(i >= 0 && i <= 2) {
    if(i == 0) {
        System.out.println("(a)");
    } else {
        System.out.println("(b)");
    }
} else if(i == 3) {
    System.out.println("(c)");
} else {
    System.out.println("(d)");
}
```

Aufgabe 2.2: `switch` in `if` umwandeln (0.5)

Wandeln Sie die folgende `switch`-Anweisung in geschachtelte `if`-Anweisungen mit gleicher Semantik um. Beachten Sie, dass es bei `case`-Klauseln, die nicht durch `break;` abgeschlossen werden, zu einem *Fall-Through* kommt:

```
int i = /* hier sollte ein Wert stehen */;

switch(i) {
    case 0: i++;
    case 1: i--;
    case 2: i += 2;
            break;
    case 3: i += 4;
    default: i = 0;
}
```

Sie dürfen arithmetische Ausdrücke beliebig vereinfachen.

Aufgabe 2.3: Schleifen umwandeln (1)

Wandeln Sie die folgende `for`-Schleife in (a) eine `while`-Schleife und (b) eine `do-while`-Schleife um:

```
Scanner sc = new Scanner(System.in);

int count = 0;

for ( boolean oneMoreTime = true ;
      oneMoreTime ;
      oneMoreTime = sc.next().equals("ja") ) {
    count ++;

    System.out.println("Noch einmal?");
}

System.out.println(count);
```