Programmierung A/B (in Java) – Übung 6

Thema: *Polymorphismus, Felder und Interfaces*

Lernziele: *Sie sollen lernen, sich in eine gegebene, komplexere Codestruktur einzuarbeiten und dabei die durch Polymorphismus bedingten Ausgaben zu verstehen. Weiterhin sollen Sie sich diese Eigenschaft durch eine eigene Implementierung selbst zunutze machen.*

Aufgabe 1: Gegeben seien folgende Klassen und Interfaces:

```
public interface I { public void a(int v1); }

public abstract class X implements I {
	protected int v1=70;
	public int v2=20;

	public void a(int v1) { v2 = v1; }

	public void print() {
		System.out.println("X: v1=" + v1 + " v2=" + v2);
	}
}

public class Y extends X {
	int v3 = 33;

	public void b(int v2) { v3 = this.v2; v1 = v2; }

	public void print() {
		System.out.println("Y: v1=" + v1 + " v2=" + v2 + " v3=" + v3);
	}
}

public class Z extends Y {
	public static int v2 = 8;
	private int v3 = 40;

	public void a(int v1) { super.a(v1); this.v1 = v1;}

	public void print() {
		System.out.println("Z: v1=" + v1 + " v2=" + v2 + " v3=" + v3);   }
}
```

```
public static void main(String[] args) {

      //ERGAENZUNGEN HIER
}
```

Verschaffen Sie sich zunächst einen Überblick über die Klassenhierarchie und die in den Klassen vorhandenen Felder. In den Klassen werden Methoden der Superklasse überschrieben und einige Felder durch gleiche Bezeichner verdeckt. Dies hat natürlich Auswirkungen auf das Programmverhalten, abhängig davon, um welchen Typ es sich handelt. Nehmen Sie an, dass an obiger Ergänzungsstelle der main-Methode jeweils eine der folgenden Anweisungsgruppen steht und geben Sie jeweils die Ausgabe an (Begründen Sie ihre Ausgabe, sollte der Code nicht kompilieren).

a)
```
Y y = new Y();
Z z = new Z();
y.print();
z.print();
```

b)
```
Y y = new Y();
y.b(42);
y.print();
```

c)
```
X x = new X();
x.a(27);
x.print();
```

d)
```
I i = new Y();
i.a(7);
i.print();
```

e)
```
X x = new Z();
x.a(55);
x.print();
```

f)
```
Y y = new Z();
System.out.println("v2=" + y.v2);
y.b(111);
y.print();
```

Aufgabe 2: In dieser Aufgabe sollen Sie ein Programm erstellen, mit Hilfe dessen Sie auf einem gegebenen zweidimensionalen Array vom Typ int verschiedene Operationen ausführen können. Zu jeder Operation soll eine eigene Klasse erstellt werden, in welcher die entsprechenden Veränderungen am Array durchgeführt werden. Die Struktur soll dabei modular angelegt sein, sodass Sie später durch Hinzufügen weiterer Klassen die Funktionalität einfach erweitern können. Führen Sie zur Realisierung die folgenden Schritte aus:

a) Das Operationsinterface

Erstellen Sie zunächst ein Interface *IArrayOperation*. Dieses Interface soll später von allen konkreten Klassen implementiert werden, die Operationen auf dem Array durchführen sollen. Dazu wird eine Methode *arrayOperation* benötigt, die ein zweidimensionales int-Array als Parameter akzeptiert und keinen Rückgabetyp verwendet. Innerhalb dieser Methode wird das Array verändert. Außerdem soll das Interface noch die parameterlose Methode *getDescription* bereitstellen, welche eine kurze Beschreibung der von einer Klasse ausgeführten Operation als String zurückgibt.

b) Die Koordinationsklasse

Die Klasse *ArrayOperationCoordination* soll ein zu veränderndes Array als Member speichern und die Ausführung aller Operationen in der Konsole protokollieren. Im Konstruktor soll das zu verändernde Array übergeben werden, welches von der Klasse als Member gespeichert wird. Gehen Sie davon aus, dass immer ein vollständig initialisiertes Array der Dimensionen m*n übergeben wird. Geben Sie zudem bei Initialisierung den aktuellen Inhalt des Arrays aus (siehe folgenden Absatz).

Des weiteren soll *ArrayOperationCoordination* eine parameterlose Methode *printArray* ohne Rückgabewert besitzen, welche den aktuellen Inhalt des Arrays formatiert in die Konsole ausgeben soll. Verwenden Sie dazu Tabulatoren zwischen den Werten (`"\t"`). Eine Beispielausgabe wäre die folgende:

```
|	11	27	8	|
|	3	18	42	|
```

Die zweite zu implementierende Methode *performOperation* soll als Parameter ein Objekt vom Typ *IArrayOperation* akzeptieren und ebenfalls keinen Rückgabewert besitzen. Innerhalb dieser soll die *arrayOperation* Methode des übergebenen Parameters auf dem als Member gespeicherten Array ausgeführt werden. Geben Sie zur Protokollierung des Vorgangs erst eine Beschreibung der durchgeführten Operation und anschließend den Inhalt des Arrays nach der Operation in die Konsole aus.

Zuletzt soll mit der Methode *getArray* eine Kopie des aktuellen Arrays erstellt und zurückgegeben werden, sodass es jederzeit zwischen mehreren Operationen möglich ist, mit den Zwischenergebnissen zu arbeiten.

c) Die Operationen

Nun sollen Sie einige Klassen erstellen, welche das Interface *IArrayOperation* implementieren und folgende Operationen realisieren:

- Zu jedem Element des Arrays soll der Betrag berechnet werden.
- Alle Elemente des Arrays sollen Modulo x gerechnet werden, wobei x als Variable über den Konstruktor bei Initialisierung festzulegen ist.
- Alle Elemente, deren Position innerhalb des Arrays y Stellen oder weniger von den Rändern entfernt sind, sollen auf 0 gesetzt werden, wobei y als Variable über den Konstruktor bei Initialisierung festzulegen ist.

d) Testen der Implementierung

Schreiben Sie eine Testklasse, in dessen main-Methode ein 5x5 Elemente großes int-Array mit zufälligen Werten initialisiert wird und testen Sie Ihre implementierten Operationen mit verschiedenen Parametern.