Institut für Informatik
Prof. Dr.-Ing. Elke Pulvermüller
Dipl.-Systemwiss. Mathias Menninghaus

Universität Osnabrück, 25.04.2017
http://www-lehre.inf.uos.de/~binf
Testat bis 10.05.2017, 14:00 Uhr

Übungen zu Informatik B

Sommersemester 2017

Blatt 3

Diese Aufgabenblatt behandelt Vorlesungsstoff über einen Zeitraum von zwei Wochen. Die Aufgaben 2 und 3 werden erst mit dem Vorlesungsstoff aus der ersten Maiwoche lösbar sein. Die nächsten Übungen finden am 04.05. statt.

Aufgabe 3.1: `equals` und `hashCode` (25 Punkte)

Lesen Sie die `javadoc`-Dokumentation zu den Methoden `equals` und `hashCode` der Klasse `Object`. Betrachten Sie die Klassen `Student` und `Person` und beurteilen Sie anhand einer eigenen separaten Testklasse mit aussagekräftiger Ausgabe, ob die Methoden `equals` und `hashCode` in diesen beiden Klassen korrekt implementiert wurden. Achten Sie dabei auch auf die Beziehungen zwischen Unter- und Oberklasse. Geben Sie für jeden Fehler, den Sie entdecken, mindestens einen Lösungsvorschlag an.

Aufgabe 3.2: Dynamisches Binden (15 Punkte)

Laden Sie die Dateien `Bird.java`, `Parrot.java`, `Dodo.java` und `Aviary.java` herunter, kompilieren Sie sie und führen Sie die Klasse `Aviary` aus. Erklären Sie Ihrem Tutor schriftlich mit Hilfe der Fachbegriffe aus dem Skript jede Zeile der Ausgabe anhand des Quellcodes.

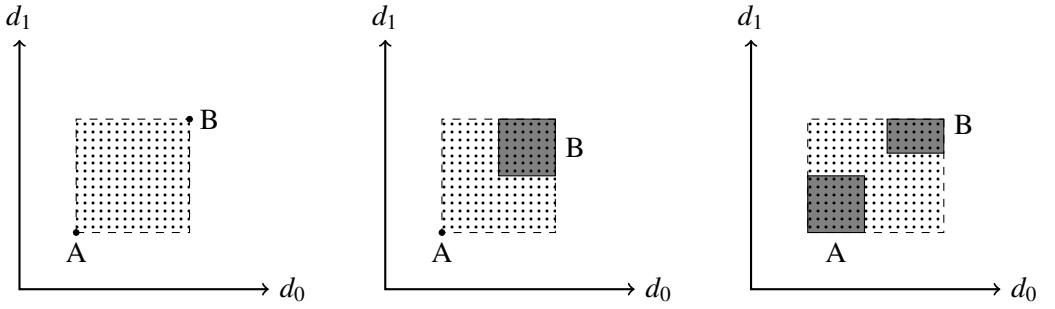
Aufgabe 3.3: Vererbung (60 Punkte)

In dieser Aufgabe sollen Sie eine kleine Geometrie-Bibliothek implementieren. Betrachten Sie dazu die abstrakte Klasse `Geometry` und Ihre javadoc-Dokumentation. Diese besitzt folgende Methoden:

- `dimensions()` - Gibt die Anzahl der Dimensionen dieser `Geometry` zurück. Eine `Geometry` hat mindestens **2** Dimensionen.
- `encapsulate(Geometry other)` - Umgibt diese und die übergebene `Geometry` mit der gleichen Anzahl an Dimensionen mit einer neuen `Geometry` und gibt diese zurück. Die neue `Geometry` muss einerseits eine minimale Ausdehnung haben und andererseits diese und die übergebene `Geometry` vollständig enthalten. Wenn zwei `Geometry` - Instanzen eine unterschiedliche Anzahl an Dimensionen haben, wird `null` zurückgeliefert.
- `volume()` - Gibt das Volumen einer `Geometry` als `double` zurück. Im zweidimensionalen Fall ist dies die Fläche einer `Geometry`.

Erweitern Sie die Klasse `Geometry` nun um die folgenden Klassen.

- `Point2D` - Ein zweidimensionaler Punkt der mit zwei `double` - Werten erzeugt werden kann.
- `Point` - Repräsentiert einen *n*-dimensionalen Datenpunkt, der mit einer variablen Parameterliste von `double` - Werten erzeugt werden kann.
- `Rectangle` - Ein Rechteck, das mit zwei Objekten vom Typ `Point2D` erzeugt werden kann.
- `Volume` - Ein `Volume` kann durch zwei *n*-dimensionale Punkte erzeugt werden, die ein rechtwinkliges Volumen aufspannen, dessen Kanten alle achsenparallel verlaufen.



(a) Umfassung von zwei Punkten.

(b) Umfassung von einem Punkt und einem Rechteck.

(c) Umfassung von zwei Rechtecken.

Abbildung 1: Beispiele für die `encapsulate(Geometry)` Methode von `Geometry`.

Implementieren Sie in allen `Geometry` - Typen das Interface `Comparable` mit der Methode `compareTo(Object o)`. Diese soll zwei Instanzen vom Typ `Geometry` immer anhand ihres Volumens miteinander vergleichen. Bei der Implementierung können Sie die Warnung des Compilers `Comparable is a raw type. References to generic type Comparable<T> should be parameterized` ignorieren.

Ordnen Sie diese Klassen in der Vererbungshierarchie möglichst geschickt an, um sich unnötige Arbeit zu ersparen. Achten Sie immer auf *Information Hiding* und machen Sie so wenige Datenfelder und Methoden wie möglich sichtbar. Sorgen Sie dafür, dass keine inkonsistenten Instanzen entstehen können.

Schreiben Sie außerdem eine separate Testklasse, die automatisch jede der implementierten Funktionen testet.

Hinweis

Sie können sich viel Arbeit ersparen, in dem Sie erst überlegen, wie der eindimensionale Fall aussieht und diesen dann auf n Dimensionen übertragen.