Hochschule Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Prof. Dr. Peter Becker

1. Testklausur — Sommersemester 2013
Datenstrukturen und Algorithmen
24. Mai 2013

Bevor Sie mit der Bearbeitung dieser Klausur beginnen, lesen Sie bitte folgende Hinweise. Diese Hinweise sind bei der Bearbeitung zu beachten.

1. Prüfen Sie die Vollständigkeit Ihres Exemplars. Es sollte

   - dieses Hinweisblatt und
   - vier Aufgaben auf vier Blättern

   umfassen.

2. Tragen Sie auf jedem Lösungsblatt oben an den vorgesehenen Stellen Ihren Namen und Ihre Matrikelnummer ein. Blätter ohne diese Angaben werden nicht bewertet.

   Hinter den Aufgaben ist jeweils hinreichend Platz für die Lösungen freigelassen. Reicht der Platz nicht aus, benutzen Sie die Rückseiten, wobei die Zuordnungen von Lösungen zu Aufgaben deutlich erkennbar sein müssen.

3. Geben Sie dieses Deckblatt zusammen mit den Aufgabenstellungen und den Lösungen sowie alles weitere beschriebene Papier ab.

4. Zugelassene Hilfsmittel: keine

5. Mit ≥ 30 Punkten haben Sie die Klausur bestanden.

6. Ergebnis (bitte nichts eintragen):

1 (15)	2 (15)	3 (10)	4 (20)	$\sum_{Punkte}$ (60)

Viel Erfolg!

Name: **Matrikel:**

Aufgabe 1 (15 Punkte)

- Ein Versandhandelsgeschäft verkauft Bücher und DVDs. Nur solche Objekte sollen erzeugt werden können.
- Sowohl Bücher als auch DVDs haben eine Bezeichnung, einen Nettoverkaufspreis, einen Bruttoverkaufspreis und einen Mehrwertsteuersatz.
- Für Bücher ist zusätzlich der Autor bekannt, für DVDs der Regisseur.
- Der Bruttoverkaufspreis errechnet sich aus dem Nettoverkaufspreis plus Mehrwertsteuer.
- Für Bücher fallen stets 7% Mehrwertsteuer (vom Nettoverkaufspreis) an, für DVDs 19%.

Definieren Sie Java-Klassen, die die oben genannten Punkte angemessen umsetzen. Stellen Sie dabei Methoden bereit, um

- die Bezeichnung zu ermitteln,
- den Nettoverkaufspreis zu setzen,
- den Nettoverkaufspreis zu ermitteln,
- den Bruttoverkaufspreis zu ermitteln,
- den Mehrwertsteuersatz zu ermitteln und
- Bücher bzw. DVDs als Zeichenkette zu repräsentieren (Methode `toString()`).

Name: **Matrikel:**

Aufgabe 2 (4+4+4+3=15 Punkte)

Gegeben sei die folgende Klasse `Counter`:

```
public class Counter {

    private int value = 0;

    public void increment() { this.value++; }

    public int get() { return this.value; }
}
```

Für eine Anwendung benötigen wir die folgende Funktionalität: Immer wenn für einen Counter die Methode `increment()` aufgerufen wird, soll ein anderes Objekt (ein sogenannter *Observer*) darüber benachrichtigt werden. Der Observer soll nicht aus einer bestimmten Klasse stammen müssen. Außerdem muss sich der Observer vorher beim Counter registrieren.

(a) Definieren Sie eine Schnittstelle für die Benachrichtigung des Observers.

(b) Implementieren Sie die Schnittstelle aus (a) beispielhaft.

(c) Ändern Sie die Implementierung von `Counter` so ab, dass sich ein Observer für die Benachrichtigung registrieren kann und im Falle eines Aufrufs von `increment()` auch benachrichtigt wird.

(d) Wie würden Sie (c) lösen, wenn Sie den Quelltext von Counter nicht anpassen könnten (z.B. weil die Klasse `Counter` aus einer fremden Bibliothek stammt).

Name: **Matrikel:**

Aufgabe 3 (2+4+4=10 Punkte)

(a) Nennen Sie eine geprüfte (ausgenommen `Exception`) und eine ungeprüfte Exception-Klasse.

(b) In der Methode `foo(int a)` soll die geprüfte Exception `ExceptionX` ausgelöst werden, wenn der Methodenparameter `a` negativ ist. Skizzieren Sie den Aufbau der Methode `foo(int a)`.

(c) Die Methode `foo(int a)` soll innerhalb der Methode `bar()` aufgerufen werden. Stellen Sie dar, welche Möglichkeiten Sie in `bar()` haben, um mit der Exception, die von `foo` möglicherweise ausgelöst wird, umzugehen.

Name: **Matrikel:**

Aufgabe 4 (20 Punkte)

Wir wollen eine generische Liste implementieren, in der die enthaltenen Objekte aufsteigend *sortiert* verwaltet werden. Das erste Element der Liste ist also stets ein kleinstes Element, das letzte ein größtes Element. Die sortierte Liste soll *einfach verkettet* sein und die folgenden Methoden bereitstellen:

- `insert`: Füge ein neues Element in die Liste ein. Das Element muss gemäß der Sortierung an der richtigen Stelle in der Liste platziert werden.
- `deleteMin`: Lösche das kleinste Element.
- `getMin`: Ermittle das kleinste Element der Liste.
- `print`: Gib die komplette Liste sortiert aus.

Geben Sie die Klasse komplett an (Klassenrahmen, innere Klassen, Instanzvariablen, Methodenköpfe und Implementierungen).