Nachholklausur zur Vorlesung: Einführung in die objektorientierte Programmierung mit Java WS1415

Alexander Bazo 10. April 2015

Allgemeine Hinweise

1. Die Bearbeitungszeit beträgt 60 Minuten. Sie könne 50 Punkte erreichen.
2. Verwenden Sie für das Lösen der Aufgaben die Programmiersprache Java. Alle gegebenen Code-Fragmente sind ebenfalls in Java verfasst. Halten Sie sich an die bekannten Regeln zur Codequalität.
3. Schreiben Sie Ihren Namen, Vornamen, Studiengang und Ihre Matrikelnummer leserlich unten auf jedes Angabenblatt bevor Sie mit der Bearbeitung beginnen! Blätter ohne diese Angaben werden nicht gewertet. Wenn Sie die Rückseite eines Blattes verwenden, notieren Sie dies bitte auf der Vorderseite. Kennzeichnen Sie eindeutig, zu welcher Aufgabe eine Lösung gehört.
4. Benutzen Sie keine Bleistifte, keine rotschreibenden Stifte und kein TippEx (oder ähnliche Produkte).
5. Die Klausur ist als *Closed Book*-Klausur angelegt. Sie dürfen keine mitgebrachten Quellen oder Notizen zur Bearbeitung der Aufgaben verwenden. Technische Hilfsmittel sind ebenfalls nicht erlaubt.
6. Wenden Sie sich bei Unklarheiten in den Aufgabenstellungen immer an die Klausuraufsicht (Hand heben). Hinweise und Hilfestellungen werden dann, falls erforderlich, offiziell für den gesamten Hörsaal durchgegeben. Aussagen unter vier Augen sind ohne Gewähr.
7. Geben Sie keine mehrdeutigen (oder mehrere) Lösungen an. In solchen Fällen wird stets die Lösung mit der geringeren Punktzahl gewertet. Eine richtige und eine falsche Lösung zu einer Aufgabe ergeben also null Punkte.

Teil 1: Theorie

1) Vergleiche (2 Punkte)

Nennen Sie kurz - ein Wort - die Art des Vergleichs, die mit den folgenden Operatoren ausgeführt wird.

```
<=
```

```
!=
```

```
==
```

```
>
```

2) Klassen und Objekte (4 Punkte)

Beschreiben Sie kurz die Begriffe *Klasse* und *Objekt* und erläutern Sie deren Zusammenhang.

3) Softwareengineering (4 Punkte)

Beschreiben Sie kurz den *Top-Down-Ansatz* und erläutern Sie in diesem Zusammenhang das Prinzip der *Decomposition*.

Teil 2: Codeverständnis

4) Codeanalyse (5 Punkte)

Geben ist die Methode `doStuff(int[] numbers)`. Beschreiben Sie kurz den Aufbau der Methode und nennen Sie anschließend den wahrscheinlichen Einsatzzweck. Gehen Sie davon aus, dass der gegebene Code korrekt ist.

```
1 public int doStuff(int[] numbers) {
2   int min = numbers[0];
3   for (int i = 1; i < numbers.length; i++) {
4     if(numbers[i] < min) {
5       min = numbers[i];
6     }
7   }
8   return min;
9 }
```

5) Fehler finden (5 Punkte)

Finden Sie die Konventionsverstöße, Syntaxfehler und Bugs in der folgenden Methode. Markieren Sie die fehlerhaften Stellen im Code und beschreiben Sie den jeweiligen Fehler kurz unter Angabe der Zeilennummer. Schlecht gewählte Bezeichner für Methoden und Variablen sowie fehlerhafte Einrückungen zählen nicht als Fehler. Im Code sind fünf Fehler zu finden.

```
1  public class StringReverser {
2
3          private static final String defaultResult = "";
4          private String baseString;
5
6          public StringReverser(String baseString) {
7                  baseString = baseString;
8          }
9
10         public int getBaseString() {
11                 return baseString;
12         }
13
14         public String getReversedString() {
15                 String result = defaultResult;
16                 for(int i=baseString.length(); i >= 0; i--) {
17                         result += baseString[i];
18                 }
19                 return result;
20         }
21
22 }
```


Name, Studiengang & Matrikelnummer:

Teil 3: Code implementieren

6) Film-Datenbank (15 Punkte)

Geben sind die abstrakte Basisklasse `Item` sowie das Enum `Type`. Entwerfen Sie mit diesen Vorgaben weitere Klassen um ein Programm zur Verwaltung von Filmen zu entwerfen. `Item` stellt den Basistyp für alle Filme in der Datenbank dar. Für die unterschiedlichen Medien-Typen werden eigenständige Klassen auf der Basis dieser Elternklasse erstellt. Gehen Sie davon aus, dass der gegebene Code korrekt ist.

```
public abstract class Item {
        private int year;
        private String title;
        private Type type;
        public Item(int year, String title) {
                this.year = year;
                this.title = title;
                this.type = Type.MISC;
        }
        public int getYear() {
                return year;
        }
        public String getTitle() {
                return title;
        }
        public Type getType() {
                return type;
        }
        public void setType(Type type) {
                this.type = type;
        }
}
```

```
public enum Type {
  DVD,
  VHS,
  MISC
}
```

a) Implementieren Sie zwei Klassen zur Verwaltung von *DVDs* und *VHS-Kassetten*. Beide Klassen erben von `Item`. `DVD` verfügt über ein zusätzliches, öffentliches Attribut *extras* vom Typ `String`. In der Klasse `VHS` wird über das öffentliche Attribut *rewound* (`boolean`) angegeben, ob die Kassette zurückgespult ist. Für beide Klassen werden die Attribute im Konstruktor durch entsprechende Parameter initialisiert. Zusätzlich wird jeweils das geerbte Attribut `type` auf den passenden Wert gesetzt.

b) Entwerfen Sie eine öffentliche Methode `filterByYear`, die ein Array vom Typen `Item` als Parameter übergeben bekommt. Anhand zweier `int`-Parameter werden aus diesem Array alle Items herausgefiltert, die zwischen den angegebene Jahren (`int`-Parameter) erschienen sind. Das Resultat wird als `ArrayList<Item>` zurückgegeben. Sie müssen nur die Methode notieren und müssen keine umschließende Klasse angeben.

Name, Studiengang & Matrikelnummer:

7) Anagramm-Checker (15 Punkte)

Implementieren Sie eine Klasse `AnagrammChecker`. Diese verfügt über eine öffentliche, statische Methode `isAnagramm`, die überprüft, ob zwei als Parameter übergebene Wörter Anagramme voneinander sind. Ein Anagramm ist ein Wort, das aus den Zeichen eines anderen Wortes gebildet werden kann. *Mehl* ist ein Anagramm von *Helm*.

Die beiden Parameter werden als Zeichenketten vom Typ *String* an die Methode übergeben. Nach der Überprüfung der beiden Wörter gibt die Methode einen boolschen Wahrheitswert zurück: Handelt es sich bei den beiden Wörtern um Anagramme, wird `true` zurückgegeben. Im anderen Fall wird `false` zurückgegeben. Ihre Methode muss mit Strings beliebiger Länge funktionieren. Dabei gelten zwei Wörter nur dann als Anagramm, wenn sie *1 zu 1* aus den Zeichen des jeweilig anderen Wortes (ohne Auslassen von Zeichen) gebildet werden können.

Hinweis: Über die Methode `char charAt(int index)` der `String`-Klasse können Sie auf ein Zeichen an einer bestimmten Stelle des Wortes zugreifen. Die Methode `int indexOf(char ch)` der selben Klasse überprüft, ob ein übergebener Buchstabe (`ch`) in dem String vorhanden ist. Im positiven Fall wird die Position (*index*) des Buchstaben als `int`-Wert zurückgeben, im negativen Fall -1.

Einen Ausschnitt eines Strings können Sie über die `String`-Methode `String substring(int beginIndex, int endIndex)` extrahieren. Zurückgegeben werden alle Zeichen von der Position `beginIndex` (inklusive) bis zur Position `endIndex` (exklusive).

Name, Studiengang & Matrikelnummer: