Klausur Programmierung I

Universität Trier
Fachbereich IV-Informatik

Wintersemester 2012/2013

Prof. Dr. Bernd Walter und **Peter Birke**

14.02.2013
Bearbeitungszeit: 120 Minuten

Name	_____
Matr.Nr	_____
Punkte Gesamt	_____ von 40
Punkte Note	_____

Beachten Sie die Rückseite

Hiermit bestätige ich, dass meine obigen Angaben richtig sind und dass ich entsprechend der Prüfungsordnung des von mir belegten Studienganges zur Teilnahme an dieser Klausur (Programmierung I, Wintersemester 2012/2013) berechtigt bin. Sollte diese Berechtigung nicht vorliegen, bin ich damit einverstanden, dass meine Teilnahme an der Klausur nicht gewertet wird.

Trier, den _____

Unterschrift: _____

Auszufüllen von Studierenden mit Studienziel Bachelor oder Master
Handelt es sich bei dieser Prüfung um den 3. Versuch den Leistungsnachweis Programmierung I zu erlangen?

Ja: □
Nein: □

Regeln

- Schreiben Sie in jede Datei als geeigneten Kommentar Ihren Namen.
- Fügen Sie bei allen Aufgaben Ihren Quellcode an den mit TODO gekennzeichneten Stellen ein.

0 Pkt.

Programmierung I (40 **Punkte sind zu erreichen**)

(4 Pkt.) **1.** Schreiben Sie eine Methode `public static int kgV(int num0, int num1)`, die das kleinste gemeinsame Vielfache der als Argument übergebenen Zahlen berechnet. Testen Sie, dass es sich bei der Eingabe um positive Zahlen handelt. Das kleinste gemeinsame Vielfache zweier natürlicher Zahlen x und y ist definiert als

$$kgV(x, y) = \min\{z \mid x \text{ teilt } z \wedge y \text{ teilt } z\}$$

Für diese Aufgabe steht Ihnen bereits im Ordner `src/aufgabe1` das Framework in `ExamMath.java` und eine Möglichkeit zum Testen Ihrer Methode in `ExamMathTest.java` zur Verfügung.

(6 Pkt.) **2.** Gegeben sei die folgende Klasse, die ein Konto repräsentiert:

```
public class Account {
  private int id;
  private double balance;
  private String customer;

  public Account(int id, double balance, String customer) {
    this.id = id;
    this.balance = balance;
    this.customer = customer;
  }

  public final void deposit(double d) {
    if (d > 0) this.balance += d;
  }

  public void withdraw(double d) {
     if (d > 0) this.balance -= d;
  }
}
```

Schreiben Sie eine Unterklasse `CheckingAccount`, deren Objekte zusätzlich ein festgeschriebenes Überziehungslimit `double overdraft` nicht überschreiten dürfen. Das zusätzliche Attribut soll wie `id, balance` und `customer` im Konstruktor mit Werten belegt werden. Die Methode `withdraw` soll allerdings, sobald ein Abheben das Limit überschreiten sollte, eine Ausnahme `IllegalArgumentException` werfen.
Sie dürfen keine Änderungen an der Klasse `Account` vornehmen.

(7 Pkt.) **3.** In dem Ordner `src/aufgabe3` stehen Ihnen die Klassen `Project` und `ProjectTest` zur Verfügung. Ein Projekt wird dabei durch einen Arbeitstitel näher beschrieben. Dieser wird im Konstruktor übergeben und kann mit Hilfe der Methode `getTitle()` ausgelesen werden.

(a) (3 pts) Schreiben Sie eine Methode `String reverseWord(String arg)`, die die übergebene Zeichenkette umkehrt. Definieren Sie die Methode so, dass Sie an keine konkrete Instanz eines Projektes gebunden ist.

(b) (4 pts) Erweitern Sie Ihre Klasse um eine Methode `String reverse()`, die eine Zeichenkette zurückgibt, in der die einzelnen Wörter des Titels eines Projekts umgedreht wurden.

17 Pkt.

(12 Pkt.) **4.** Die bereitgestellten Klassen `IntElem`, `SortedList` und `SortedListTest` aus dem Ordner `src/aufgabe4` realisieren aufsteigend sortierte, doppelt verkettete Listen. In der Datei `SortedListTest.java` steht mit der `main`-Methode ein Testoprogramm zur Verfügung, mit `IntElem.java` werden die Elemente der Liste umgesetzt. Die Klasse `SortedList` repräsentiert mit Hilfe der Instanzvariablen

```
private IntElem start;
private IntElem ende;
```

eine doppelt verkettete Liste. `start` verweist dabei auf das erste Listenelement, und `ende` auf das letzte. Ergänzen Sie in der Datei `SortedList.java` die folgenden Methoden mit der gewünschten Funktionalität:

(a) (5 pts) `IntElem getPrev(int value)` : Gibt das letzte Listenelement zurück, dessen Wert noch echt kleiner als der Parameter `value` ist.

(b) (7 pts) `void add(int valueToInsert)` : Fügt den als Argument übergebenen Wert in die Liste ein. Beachten Sie, dass die Sortierung nach der Einfügeoperation erhalten bleibt. Hinweis: Sie können die zuvor implementierte Methode `getPrev` evtl gebrauchen.

(11 Pkt.) **5.** Um eine beliebige Menge von ganzzahligen Werten `int` zu verwalten, sollen Sie eine Klasse `UArray` implementieren. Diese soll die zu speichernden Zahlen in einem Array ablegen. Da ein Array nur eine maximale Größe hat, sollen Sie innerhalb der Klasse dafür sorgen, dass wenn beim Einfügen einer Zahl diese nicht mehr gespeichert werden kann, das Feld in ein doppelt so großes kopiert wird.

```
public class UArray {
 private int[] internalArray;
 private int size;

 public UArray(int initCapacity) {
   internalArray = new int[initCapacity];
   size = 0;
 }

 void ensureCapacity() {
   // TODO Implementieren
 }

 public void add(int arg) {
   ensureCapacity();
   internalArray[size++] = arg;
 }
}
```

Stellen Sie zusätzlich die folgenden Methoden zur Verfügung:

(a) (4 pts) Eine Methode `void ensureCapacity()`, die testet, ob das interne Array ein weiteres Element aufnehmen kann. Wenn dies nicht der Fall ist, soll die Methode die Werte des aktuellen internen Arrays in ein neues, doppelt so großes Array kopieren, und dieses künftig innerhalb der Klasse verwenden. Diese Methode soll nur innerhalb der Klasse zur Verfügung stehen.

(b) (1 pt) Die Methode `int getSize()` liefert die Anzahl der aktuell gespeicherten Werte zurück, während

(c) (1 pt) `int getCapacity()` die Anzahl der zum aktuellen Zeitpunkt speicherbaren Elemente zurückgibt.

(d) (2 pts) Mittels `int get(int index)` geben Sie den `int` Wert an der Position `index` zurück. Wenn der übergebene Parameter außerhalb des gültigen Bereichs liegt, soll die Methode eine Ausnahme, nämlich eine `IndexOutOfBoundsException`, werfen.

(e) (3 pts) `int indexOf(int arg)` gibt die erste Position zurück, an der der Wert `arg` gespeichert ist. Wenn der gesuchte Wert nicht in diesem Objekt enthalten ist, soll die Methode -1 zurückgeben.

23 Pkt.