technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Iman Kamehkhosh, Marcel Preuß, Henning Timm

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2013/14

Übungsblatt 11

Bearbeitungszeit:
27.–31.01.2014

Auf diesem Übungsblatt finden Sie ein Punktesystem.
Die Punkteverteilung entspricht in etwa einer Klausur mit insgesamt 100 Punkten.
Es soll Ihnen helfen, die Schwierigkeit und den Zeitaufwand der jeweiligen Aufgabe einzuschätzen.

Aufgabe 11.1 – Aufgaben von Blatt 10

Bitte lösen Sie zunächst die regulären Aufgaben von Blatt 10 , die Sie noch nicht bearbeitet haben.

Aufgabe 11.2 – Listen

(15 Punkte)

Für die folgenden Aufgaben benötigen Sie die Klasse `Element`. Benutzen Sie nicht die Version aus der Vorlesung, sondern folgende vereinfachte Variante.

```
public class Element {
  public int wert;
  public Element weiter;
  public Element(int wert) {
    this.wert = wert;
    this.weiter = null;
  }
}
```

a) Programmieren Sie eine Klasse `Liste`, die ein privates Attribut namens `kopf` vom Typ `Element` besitzt. Der Konstruktor soll keine Parameter haben und dieses Attribut auf `null` setzen. Fügen Sie eine main-Methode hinzu, in der sie die Klasse nach jeder der folgenden Änderungen testen. (2 Punkte)

b) Fügen Sie der Klasse eine Methode `einfuegenVorn` hinzu, die ein `Element` mit dem übergebenen ganzzahligen Wert erstellt und dieses am Anfang der Liste einfügt. (3 Punkte)

c) Fügen Sie eine Methode `istLeer` hinzu, die zurückgibt, ob die Liste leer ist (`true` wenn leer, `false` wenn nicht). (1 Punkt)

d) Fügen Sie eine Methode `anzahl` hinzu, die zurückgibt, wie viele Elemente die Liste besitzt. (2 Punkte)

e) Fügen Sie eine Methode `letztesElement` hinzu, die das letzte Element der Liste zurückgibt. Es soll *nicht* der Wert, sondern eine Referenz auf das Element zurückgegeben werden. Ist die Liste leer, so wird `null` zurückgegeben. (3 Punkte)

f) Fügen Sie der Klasse `Liste` eine Methode `gibAlleAus` hinzu, die die Werte aller Elemente der Liste in der gespeicherten Reihenfolge ausgibt. Die Liste und die Werte sollen dabei *nicht* sortiert werden. Die einzelnen Werte sollen durch Leerzeichen getrennt werden. (4 Punkte)

Aufgabe 11.3 – Schlüsselwörter (6 Punkte)

a) Erklären Sie in eigenen Worten die Deklaration der `main`-Methode: (2 Punkte)
`public static void main(String[] args)`

b) Die Kreiszahl $\pi \approx 3,14159$ ist eine mathematische Konstante, die in der Java-internen Klasse `Math` deklariert ist, und die aus jeder anderen Klasse über `Math.PI` erreichbar ist, ohne ein Objekt der Klasse `Math` zu instanziieren. Anhand welcher Schlüsselwörter muss demnach `PI` deklariert sein? (2 Punkte)

c) Erklären Sie in eigenen Worten die Gemeinsamkeiten und Unterschiede der Schlüsselwörter `public`, `protected` und `private`. (2 Punkte)

Aufgabe 11.4 – Adressbuch – Objektorientierung all-in-one

Firmen benötigen für Testzwecke häufig große Datensätze für ihre Datenbanken. Ihre Aufgabe ist es, ein Programm zu schreiben, welches ein Adressbuch generiert, in dem eine konstante Anzahl an zufällig generierten Personendaten mit Vorname, Nachname, sowie Adresse (Straße, Hausnummer, Postleitzahl, Ort) erfasst werden. Die Personen werden in zwei Gruppen (Mitarbeiter und Kunde) eingeteilt. Ein Mitarbeiter hat ein Gehalt, wohingegen ein Kunde eine Kundennummer.

Erstellen Sie 4 Dateien, in denen Sie jeweils einen Klassenrumpf mit entsprechenden Namen vorbereiten:

- Anwendung.java
- Person.java
- Adresse.java
- Adressbuch.java

Modellieren Sie die Klassen entsprechend folgender Beschreibung:

a) **Person:**
Die Klasse `Person` enthält folgende *private* Attribute:

- Zeichenkette Vorname
- Zeichenkette Nachname
- Markierung, die angibt, ob es sich dabei um einen Kunden handelt
- Ganzzahligen Wert für eine Kundennummer
- Fließkommazahl für Gehalt, falls es sich dabei um einen Mitarbeiter handelt
- Adresse vom Typ `Adresse`

Die Klasse `Person` soll folgende Methoden unterstützen:

- Erzeugen einer Person als Mitarbeiter, der ein übergebenes Gehalt zugewiesen wird; oder als Kunde, der eine übergebene Kundennummer zugewiesen wird. Dabei wird eine vorhandene Adresse mit übergeben.
- Ausgabe der Personendaten inklusive der Adressdaten. Handelt es sich dabei um einen Mitarbeiter, so soll sein Gehalt mit ausgegeben werden; entsprechend die Kundennummer eines Kunden.
- Abfrage, ob es sich bei der Person um einen Mitarbeiter oder Kunden handelt

b) **Adresse:**
Die Klasse `Adresse` enthält folgende *private* Attribute:

- Zeichenkette Strasse
- Ganzzahligen Hausnummer
- Ganzzahligen Postleitzahl
- Zeichenkette Ort

Die Klasse `Adresse` soll folgende Methoden unterstützen:

- Erzeugen einer Adresse mit übergebenen Werten.
- Ausgabe der Adressdaten.

c) **Adressbuch:**
Die Klasse `Adressbuch` enthält folgende *private* Attribute:

- Ganzzahlige Anzahl an Datensätzen

- Array, in dem Kontakte vom Typ `Person` gespeichert werden
- Ganzzahliger Wert für die Größe des Arrays

Die Klasse `Adressbuch` soll folgende Methoden unterstützen:

- Erzeugen eines leeren Adressbuches mit fester Größe.
- Einfügen einer neuen Person in das Adressbuch. Die Anzahl an Datensätzen wird dabei um 1 erhöht. Wurde vorher ein Kontakt aus dem Adressbuch entfernt, sodass eine Lücke entstanden ist, so soll die neue Person in diese Lücke eingefügt werden. Wird die Größe des Adressbuches dabei überschritten, so soll die neue Person nicht eingefügt werden.
- Entfernen einer Person aus dem Adressbuch anhand des Vor- und Nachnamen. Die Anzahl der Kontakte soll dabei um 1 verringert werden.
- Ausgabe aller Personen, die Mitarbeiter sind.
- Ausgabe aller Personen, die Kunden sind.
- Abfrage einer Person anhand der Vor- und Nachnamen. Dabei soll ein Objekt vom Typ `Person` zurückgegeben werden, oder `null`, falls kein passender Kontakt gefunden wurde.

Nachdem unsere Datenstrukturen modelliert wurden, können wir diese in der Klasse `Anwendung` testen. Fügen Sie der Klasse `Anwendung` folgende Methode zur Generierung der Personen hinzu:

```
public static Person generierePerson(){
  String[] Vornamen =
   {"Alexander","Anna","Alina","Christian","Nicole",
    "Florian","Mia","Lisa","Sebastian","Vanessa",
    "Daniel","Oliver","Max","Katharina"};
  String[] Nachnamen =
  {"Müller","Schmidt","Mustermann","Becker",
    "Fischer","Schneider","Böhm","Hartmann"};
  String[] Strassen =
  {"Feldweg","Bürgerweg","Klammstraße","Am Feld",
    "Nordstraße","Bahnhofsalee","Beim Brandt"};
  String[] Orte =
  {"Dortmund","Bochum","Essen","Hagen","Düsseldorf",
    "Köln","Berlin","Hamburg","München","Aachen"};

  Person neuePerson = null;
  java.util.Random rd = new java.util.Random();
  int vn=rd.nextInt(14);
  int nn=rd.nextInt(8);
  int st=rd.nextInt(7);
  int or=rd.nextInt(10);
  int hn=rd.nextInt(99);
  int plz=0;
  while(plz<10101) plz=rd.nextInt(99999);
  Adresse adr = new Adresse(Strassen[st],hn,plz,Orte[or]);
  int x=0;
  while(x<1200) x=rd.nextInt(5000);
  if(rd.nextInt(10)%2==0){
     neuePerson = new Person(Vornamen[vn],Nachnamen[nn],adr,x*1.02);
  }else{
     neuePerson = new Person(Vornamen[vn],Nachnamen[nn],adr,x);
  }
  return neuePerson;
}
```

Fügen Sie der Klasse `Anwendung` eine statische `main`-Methode hinzu, die Ihr Adressbuch testet. Erstellen Sie hierzu ein neues Adressbuch fester Größe. Generieren Sie mithilfe der obigen Methode einige Personendaten und fügen Sie diese in das Adressbuch ein. Geben Sie alle Kunden aus. Suchen Sie sich einen davon heraus und entfernen Sie diesen aus dem Adressbuch. Suchen Sie im Adressbuch nach dem entfernten Kontakt.