Ludwig-Maximilians-Universität München
Institut für Informatik
PD. Dr. Peer Kröger, Dr. Andreas Züfle
Daniel Kaltenthaler, Klaus Schmid

München, 21.01.2015

Einführung in die Programmierung

WS 2014/15

Übungsblatt 13: Typsicherheit, Datenstrukturen

Besprechung: 28./30.01.2015

Ende der Abgabefrist: Dienstag, 27.01.2015 14:00 Uhr.

Hinweise zur Abgabe:
Geben Sie bitte Ihre gesammelten Lösungen zu diesem Übungsblatt in einer Datei `loesung13.zip` unter `https://uniworx.ifi.lmu.de` ab.

Wichtiger Hinweis zum Übungsbetrieb:
Am Montag, 02.02.2015, finden keine Übungen statt.
Bitte besuchen Sie stattdessen eine der Übungen am Mittwoch, 28.01.2015, oder Freitag, 30.01.2015!

Aufgabe 13-1 *Generics* **0 Punkte**

Gegeben sind folgende Klassen zur Modellierung von Krapfen mit Füllung:

```
1  public class Geschmacksrichtung {
2    public static final String[] GESCHMACKSRICHTUNGEN =
3                  {"Erdbeere", "Himbeere", "Hagebutte"};
4
5    private String geschmacksrichtung;
6
7    public Geschmacksrichtung(int geschmacksrichtung){
8      if(geschmacksrichtung < 0
9      || geschmacksrichtung >= GESCHMACKSRICHTUNGEN.length){
10       throw new IllegalArgumentException("Ungueltige Geschmacksrichtung");
11     }
12     this.geschmacksrichtung = GESCHMACKSRICHTUNGEN[geschmacksrichtung];
13   }
14
15   public String getGeschmacksrichtung(){
16     return this.geschmacksrichtung;
17   }
18 }
```

```
1  public class Fuellung {}
```

```
1  public class Marmelade extends Fuellung {
2    private Geschmacksrichtung geschmacksrichtung;
3
4    public Marmelade(Geschmacksrichtung geschmacksrichtung){
5      this.geschmacksrichtung = geschmacksrichtung;
6    }
7
8    public Geschmacksrichtung getGeschmacksrichtung(){
9      return this.geschmacksrichtung;
10   }
11 }
```

```java
public class Senf extends Fuellung {}
```

```java
public class Krapfen {
  private Fuellung fuellung;

  public void fuellen(Fuellung fuellung){
    this.fuellung = fuellung;
  }

  public Fuellung schmecktNach(){
    return this.fuellung;
  }
}
```

Diese Modellierung wird von einer Reihe von Klassen verwendet:

```java
import java.util.ArrayList;
import java.util.List;

public class Konditor {
  public List<Krapfen> krapfenMachen(int anzahl, Fuellung fuellung)
  {
    List<Krapfen> krapfenliste = new ArrayList<Krapfen>(anzahl);
    for(int i = 0; i < anzahl; i++)
    {
      Krapfen krapfen = new Krapfen();
      krapfen.fuellen(fuellung);
      krapfenliste.add(krapfen);
    }
    return krapfenliste;
  }
}
```

```java
public class Kunde {
  public Geschmacksrichtung krapfenEssen(Krapfen krapfen){
    return ((Marmelade) krapfen.schmecktNach()).getGeschmacksrichtung();
  }
}
```

```java
import java.util.List;

public class Krapfenkauf{
  public static void main(String[] args){
    Konditor grattler = new Konditor();
    List<Krapfen> krapfenliste = grattler.krapfenMachen(10, new Senf());
    Kunde kunde = new Kunde();
    for(Krapfen krapfen : krapfenliste)
    {
      if(kunde.krapfenEssen(krapfen)
              .getGeschmacksrichtung()
              .equalsIgnoreCase("ERDBEERE"))
      {
        System.out.println("Mmmmmh!");
      }
    }
  }
}
```

(a) Bei Ablauf des Programmes `Krapfenkauf` wird ein Fehler auftreten. Um was für einen Fehler handelt es sich, warum und an welcher Stelle (Klassenname, Zeilennummer) tritt er auf?

(b) Verbessern Sie die Modellierung von Krapfen mit Füllung durch Typisierung so, dass bei Ihrer Verwendung Typsicherheit zur Übersetzungszeit erreicht wird.

(Es genügt, wenn Sie hier nur die geänderten Zeilen unter Angabe von Klassenname und Zeilennummer eintragen.)

(c) Passen Sie die Klassen `Konditor`, `Kunde` und `Krapfenkauf` dieser neuen Modellierung an. Ein Kunde soll danach grundsätzlich nur Krapfen essen, die mit Marmelade gefüllt sind. Ansonsten soll sich aber im Ablauf und Ergebnis des Programmes nichts ändern.

(Es genügt, wenn Sie hier nur die geänderten Zeilen unter Angabe von Klassenname und Zeilennummer eintragen.)

Geben Sie Ihre Lösung in einer Datei `Krapfen.txt` oder `Krapfen.pdf` ab. Alternativ können Sie für die Codeaufgaben auch die entsprechenden Klassen abgeben.

Aufgabe 13-2 *Datenstrukturen* **3+4+5 Punkte**

Gegeben folgende Klasse zur Modellierung einer sortierten Liste:

```
public class SortierteListe<E extends Comparable<E>> {
  private int size;
  private Entry<E> head;

  public SortierteListe() {
    this.head = null;
  }

  public E get(int index) {
    if(index < 0 || index >= this.size) {
      throw new IndexOutOfBoundsException("Index: " + index
                                          + ", Size: " + this.size);
    }
    Entry<E> currentEntry = this.head;
    while(index > 0) {
      currentEntry = currentEntry.getNext();
      index--;
    }
    return currentEntry.getElement();
  }

  public int size() {
    return this.size;
  }

  private static class Entry<E> {
    private E element;
    private Entry<E> next;

    public Entry(E o, Entry<E> next) {
      this.element = o;
      this.next = next;
    }

    public E getElement() {
      return this.element;
    }

    public void setElement(E element) {
      this.element = element;
    }

    public Entry<E> getNext() {
      return this.next;
    }

    public void setNext(Entry<E> next) {
      this.next = next;
    }
  }
}
```

Die Liste ist aufsteigend gemäß der natürlichen Ordnung ihrer Elemente sortiert, d.h. für zwei Elemente der Liste `a` und `b` gilt: Aus `a.compareTo(b) < 0` folgt, dass das Element `a` in der Liste vor dem Element `b` steht.

Ergänzen Sie diese Klasse um folgende Methoden, wobei Sie die Eigenschaft der Sortierung nutzen, um unnötige Operationen zu sparen:

(a) Die Methode **`public int`** `indexOf(E element)` gibt den Index des ersten Vorkommens des gegebenen Elementes in der Liste zurück (beginnend mit 0 für das erste Element der Liste) oder -1, falls das Element in der Liste nicht vorkommt.

(b) Die Methode **`public void`** `add(E element)` fügt der Liste das gegebene Element hinzu, wobei die Eigenschaft der Sortierung erhalten bleiben soll.

(c) Die Methode **`public void`** `slice(E min, E max)` entfernt aus der Liste alle Elemente, die kleiner als `min` oder größer als `max` sind.

Geben Sie die Klasse `SortierteListe` ab.