RWTH Aachen
Lehrgebiet Theoretische Informatik
Frohn–Reidl–Rossmanith–Sánchez–Ströder

WS 2013/14
Übungsblatt 2
4.11.2013

Übung zur Vorlesung Programmierung

Aufgabe T4

Wir wollen die Klasse *Liste* aus der Vorlesung um folgende Funktionalitäten erweitern:

1. Die Methode **int** *size*() soll die Länge einer Liste zurückgeben.
2. Die Methode *Liste sum*(*Liste a*, *Liste b*) erzeugt aus zwei Listen eine neue Liste, welche die Summen der Elemente mit gleicher Position enhält. So ist z.B. $sum([1,2,3],[1,1,0]) = [2,3,3]$. Sind die Listen nicht gleicher Länge, so gibt die Methode *null* zurück.

Implementieren Sie beide Methoden sowohl rekursiv als auch iterativ. Sie dürfen dabei auf die Attribute der Klasse zugreifen und die Funktionen *head*(), *tail*(), *isEmpty*() und *add*(**int** *elem*) verwenden.

```
class Liste {
  private boolean empty; // Liste leer?
  private int value; // erstes Element
  private Liste rest; // restliche Elemente
  int head() {...}
  Liste tail() {...}
  boolean isEmpty() {...}
  // Erzeuge neue Liste [elem, alte Liste]
  Liste add(int elem) {...}
}
```

Lösungsvorschlag

```
  // Iterativ
  int size() {
    Liste ca = this;
    int len = 0;
    while(!ca.isEmpty()) {
      ca = ca.tail();
      len++;
    }
    return len;
  }
  // Rekursiv
  int size() {
    if(isEmpty()) {
      return 0;
```

```
    }
    return 1 + tail().size();
  }
  // Iterativ
  static Liste sum(Liste a, Liste b) {
    if(a.size() != b.size()) {
      return null;
    }
    Liste res = new Liste();
    for(int i = a.size() - 1; i >= 0; i--) {
      int pos = i;
      Liste ca = a;
      Liste cb = b;
      while(pos > 0) {
        ca = ca.tail();
        cb = cb.tail();
        pos--;
      }
      res = res.add(ca.head() + cb.head());
    }
    return res;
  }
  // Rekursiv
  static Liste sum(Liste a, Liste b) {
    if(a.size() != b.size()) {
      return null;
    }
    if(a.isEmpty()) {
      return new Liste();
    }
    Liste res = sum(a.tail(), b.tail());
    return res.add(a.head() + b.head());
  }
```

Aufgabe T5

Für das Finanzunternehmen "Megacapital Inc." sollen Sie eine Organisationssoftware entwerfen. Jeder Mitarbeiter von Megacapital hat (neben einem Namen) einen direkten Vorgesetzten. Die Firmenstruktur ist so definiert, daß zwei Mitarbeiter dann in der gleichen Abteilung arbeiten, wenn Sie einen gemeinsamen—nicht notwendigerweise direkten—Vorgesetzten haben.

Entwerfen Sie eine Klasse *Person*, welche die Gegebenheiten der Firma abbildet. Dazu soll diese Klasse Methoden anbieten, um folgende Aufgaben zu bewältigen:

- Den Namen einer Person zurückgeben
- Den direkten Vorgesetzten einer Person zurückgeben und setzen; letzters ist allerdings nur möglich, wenn die Person noch keinen Vorgesetzten hat.

- Den obersten Vorgesetzten einer Person bestimmen und zurückgeben.
- Die Abteilungen zweier beliebiger Mitarbeiter zusammenlegen; dazu soll der Chef der einen Abteilung den Chef der anderen Abteilung als direkten Vorgesetzten erhalten. Implementieren Sie diese Methode als *takeOverDepartmentOf*(*Person other*) in der Klasse *Person*, der Abteilungschef der Person *other* wird dann dem Abteilungschef der Person *this* untergeordnet.

Lösungsvorschlag

```
class Person {
  String name;
  Person directBoss;
  // Erzeugt einen neue Person ohne Vorgesetzten
  public Person(String name) {
    this.name = name;
  }
  // Weist einen Vorgesetzten zu
  public void setBoss(Person newBoss) {
    if(directBoss == null) {
      directBoss = newBoss;
    }
  }
  // Gibt den Vorgesetzten zurueck
  public Person getBoss() {
    return directBoss;
  }
  // Gibt den Abteilungsleiter zurueck
  public Person getDepartmentBoss() {
    if(directBoss == null) {
      return this;
    }
    return directBoss.getDepartmentBoss();
  }
  // Legt die Abteilung dieses Mitarbeiters mit der Abteilung
  // des anderen Mitarbeiters zusammen
  public void takeOverDepartmentOf(Person other) {
    Person myBoss = getDepartmentBoss();
    Person hisBoss = other.getDepartmentBoss();
    hisBoss.setBoss(myBoss);
  }
  // Gibt den Namen der Person zurueck
  public String getName() {
    return name;
  }
}
```

Aufgabe H4 (4+6 Punkte)[1]

Wir wollen die Klasse *Liste* aus der Vorlesung um folgende Funktionalitäten erweitern:

1. Die Methode **boolean** *contains*(**int** *element*) soll testen, ob diese Liste die Zahl *element* enthält.
2. Die Methode **boolean** *isSorted*() gibt zurück, ob die Liste aufsteigend sortiert ist.

Implementieren Sie beide Methoden sowohl rekursiv als auch iterativ.

Lösungsvorschlag

```
// Iterativ
boolean contains(int element) {
  Liste ca = this;
  while(!ca.isEmpty()) {
    if(ca.head() == element){
      return true;
    }
    ca = ca.tail();
  }
  return false;
}
// Rekursiv
boolean contains(int element) {
  if(isEmpty()) {
    return false;
  }
  return head() == element || tail().contains(element);
}
// Iterativ
boolean isSorted() {
  if(size() <= 1) {
    return true;
  }
  Liste ca = this;
  do {
    int current = ca.head();
    ca = ca.next();
    int next = ca.head();
    if(current > next) {
      return false;
    }
  } while(ca.size() > 1);
  return true;
}
// Rekursiv
boolean isSorted() {
  if(size() <= 1) {
    return true;
```

```
    }
    int current = head();
    int next = tail().head();
    if(current > next) {
      return false;
    }
    return tail().isSorted();
  }
```

Aufgabe H5 (5 Punkte)[1]

Implementieren Sie die Klasse *Person* aus Aufgabe T5. Testen Sie Ihre Implementierung anhand des folgenden Beispiels:

```
public static void main(String[ ] args){
  Person joe = new Person("Joe");
  Person hamlet = new Person("Hamlet");
  Person maggie = new Person("Maggie");
  Person julia = new Person("Julia");
  Person oliver = new Person("Oliver");
  // Maggie's department
  joe.setBoss(hamlet);
  hamlet.setBoss(maggie);
  // Oliver's department
  julia.setBoss(oliver);
  System.out.println(joe.getDepartmentBoss().getName()); // Should be Maggie
  System.out.println(oliver.getDepartmentBoss().getName()); // Should be Oliver
  // Merge
  joe.takeOverDepartmentOf(julia);
  System.out.println(oliver.getDepartmentBoss().getName()); // Should be Maggie
}
```

Abgabe zum 12.11.2013

[1] Bitte Quelltext für die Abgabe ausdrucken und zusätzlich per E-mail an den jeweiligen Tutor senden.