

## Übungsblatt 9: Software-Entwicklung 1 (WS 2017/18)

Ausgabe: 18.12.2017  
Abgabe: 8.1.2018

### **Aufgabe 0 Vorlesungsumfrage**

Füllen Sie bitte die Vorlesungsumfrage (<https://vlu.cs.uni-kl.de/teilnahme/>) aus, falls Sie dies noch nicht getan haben sollten. Wir sind besonders an konstruktiven Kommentaren interessiert.

### **Aufgabe 1 Vorbereitung auf Probeklausur**

Bereiten Sie sich auf die Probeklausur vor. Dazu ist es hilfreich alte Klausur-Aufgaben zu bearbeiten. Diese finden Sie in der Klausurausleihe des Fachbereichs Informatik unter <https://kai.cs.uni-kl.de/>. Bereiten Sie sich auch darauf vor, dass Sie in der Klausur Aufgaben handschriftlich bearbeiten müssen.

Falls Sie an der Probeklausur am 09.01.2018 teilnehmen wollen, melden Sie sich bitte bis spätestens 21.12.2017 über das STAT System dafür an.

Sie erhalten 1/3 der in der Probeklausur erreichten Punkte als Übungspunkte gutgeschrieben.

### **Aufgabe 2 Umgekehrte Polnische Notation auswerten (5 Punkte)**

Schreiben Sie ein Programm `Polnisch.java`, welches einen arithmetischen Ausdruck in umgekehrter polnischer Notation von der Standardeingabe einliest und auswertet.

Bei der umgekehrten polnischen Notation steht der Operator hinter den Operanden. Beispielsweise ergibt der Ausdruck `9 2 -` die Zahl 7 und der Ausdruck `3 9 2 - *` die Zahl 21.

Bei der Auswertung wird wie folgt vorgegangen: Der aktuelle Zustand soll mit einem Stapel (Stack) verwaltet werden. Die Eingabe wird von links nach rechts gelesen. Wenn eine Zahl eingelesen wird, so wird diese auf den Stapel gelegt. Wenn ein Operator eingelesen wird, so werden die zwei obersten Zahlen vom Stapel herunter genommen, mit dem Operator verrechnet und das Ergebnis wird auf den Stack gelegt. Am Ende der Eingabe sollte nur noch eine Zahl auf dem Stapel liegen, welche auf der Standardausgabe ausgegeben werden soll.

Verwenden Sie die Prozeduren `StdIn.isEmpty` und `StdIn.readString` zum Einlesen der Eingabe. Als Operatoren sollen `+`, `-`, `*` und `/` unterstützt werden. Strings können mit der Prozedur `Integer.parseInt` in Zahlen umgewandelt werden.

*Hinweis:* Sie müssen für die Auswertung keinen (abstrakten) Syntaxbaum konstruieren.

## Aufgabe 3 Generics (6 Punkte)

Die folgenden Programme kompilieren nicht, da sie jeweils einen oder mehrere Fehler enthalten. Erklären Sie, warum die Programme fehlerhaft sind.

a)

```
1 import java.util.*;
2 class ListOfInts {
3     public void test() {
4         List<int> list = new List<int>();
5         list.add(37);
6         System.out.println(list);
7     }
8 }
```

b)

```
1 import java.util.*;
2 class Cell<T> {
3     private T t;
4     void set(T t) {
5         this.t = t;
6     }
7     T get() {
8         return t;
9     }
10 }
11 class CellExample {
12     void test() {
13         List<Cell<T>> cells = new ArrayList<Cell<T>>();
14         cells.add(new Cell<T>());
15         System.out.println(cells);
16     }
17 }
```

c)

```
1 class Korb<T> {}
2 class Obst {}
3 class Banane extends Obst {}
4 class Mensch {
5     void essen(Korb<Obst> s) {
6         System.out.println("hmm, lecker");
7     }
8     public static void main(String[] args) {
9         new Mensch().essen(new Korb<Banane>());
10    }
11 }
```

d)

```
1 class A<S,T> {
2     public S x;
3     public T y;
4 }
5
6 class B<S> extends A<String, S> {
7     S f() {
8         return x;
9     }
10    String g() {
11        return this.x;
12    }
13    String h() {
14        return y;
15    }
16 }
```

## Aufgabe 4 Java Collection Framework (9 Punkte)

Gegeben sind die folgenden Interfaces um Bestellungen in einem Online-Handel zu modellieren:

```
interface Article {
    String getName();
    String getDescription();
    long getPrice();
}

interface Order {
    List<Article> getArticles();
    User getUser();
}

interface User {
    String getName();
}
```

Implementierungen für diese Interfaces finden Sie im Material zu dieser Übung. Ihr Code soll sich aber (bis auf Aufgabenteil e) nur auf die Interfaces beziehen und nicht auf die Klassen.

Schreiben Sie eine Klasse `Store`, welche die folgenden Prozeduren enthält.

- Die Prozedur `List<Article> search(List<Article> l, String s)` nimmt eine Liste von Artikeln `l` und gibt eine neue Liste mit den Artikeln aus der Liste zurück, welche in der Beschreibung oder im Namen den String `s` enthalten.
- Die Prozedur `long price(Order o)` nimmt eine Bestellung und berechnet deren Gesamtpreis, in dem sie die Preise der einzelnen Artikel in der Bestellung aufsummiert.
- Die Prozedur `Set<Article> neverOrdered(List<Article> articles, List<Order> orders)` nimmt eine Liste aller Artikel und eine Liste aller Bestellungen und gibt die Menge der Artikel zurück, die nie bestellt worden sind.
- Die Prozedur `Map<User, List<Article>> articlesByUser(List<Order> orders)` nimmt eine Liste von Bestellungen und gibt eine Map zurück, in der für jeden Benutzer die Liste bestellten Artikel gespeichert ist.
- Die Prozedur `List<Order> mergeOrders(List<Order> orders)` nimmt eine Liste von Bestellungen und gibt eine neue, optimierte Liste von Bestellungen zurück. In der Ergebnis-Liste sollen die Bestellungen von gleichen Benutzern zu einer einzelnen Bestellung zusammengefasst werden.

## Aufgabe 5 Weitere Übungen (freiwillig)

Zur Vorbereitung auf die Klausur kann es hilfreich sein, mit weiteren Aufgaben zu üben. Auf den folgenden Webseiten finden Sie Aufgaben, welche Sie zum Üben verwenden können. Beachten Sie aber, dass diese eventuell über den aktuellen Stand der Vorlesung hinausgehen oder gleiche Themen anders behandeln.

**CodingBat.com** Lösungen können ohne Anmeldung direkt im Browser geschrieben und getestet werden.

<http://codingbat.com/java>

**exercism.io** Verwendet ein Kommandozeilen-Programm, mit dem Lösungen hochgeladen werden können.

<http://exercism.io/languages/java/about>

**Project Euler** Enthält vor allem mathematische Probleme, welche sich programmatisch lösen lassen.

<https://projecteuler.net/archives>

**Hackerrank.com** Lösungen können direkt im Browser geschrieben und getestet werden.

<https://www.hackerrank.com/domains/java>