Informatik I **Wintersemester 2018/19**
Prof. Dr. Carsten Damm Georg-August-Universität Göttingen
Dr. Henrik Brosenne Institut für Informatik

Übungsblatt 11

Für die Klausurzulassung zählt diese Übung 100 Punkte. Trotzdem können alle Punkte dieser Übung (115 + 6 Zusatzpunkte) angerechnet werden.

E-Learning

Absolvieren Sie die Tests bis Di., 22.01.2019, 23:55 Uhr.

Die Tests sind in der Stud.IP-Veranstaltung *Informatik I* unter *Lernmodule* hinterlegt.

Sie können einen Test **nur einmal durchlaufen**. Sobald Sie einen Test starten steht Ihnen nur eine **begrenzte Zeit** zu Verfügung, um den Test zu bearbeiten.

Alle Punkte, die Sie beim Test erreichen, werden ihnen angerechnet.

ILIAS 4-Minuten-Aufgaben – 12 Punkte

Absolvieren Sie die Tests *Informatik I - ILIAS 11 Teil 1, Teil 2 und Teil 3.*
(12 Punkte)

LON-CAPA – 40 Punkte

Stacks und Queues, Postfix auswerten, Infix/Postfix

Absolvieren Sie im Test *Informatik I - LON-CAPA* die *Übung 11.*
(40 Punkte) + (6 Zusatzpunkte)

Übung

Abgabe bis Di., 22.01.2019, 18 Uhr.

Werfen Sie Ihre Lösung in den Zettelkästen Ihrer Gruppenübung. Für die Übungen im Nordbereich stehen die Zettelkästen im Sockelgeschoß (Ebene -1) **oder** auf dem Flur vor dem Seminarraum auf Ebene 0 des Instituts für Informatik.

Wenn Ihre Übung im Südbereich stattfindet, klären Sie mit Ihrem Tutor wo die Lösungen abzugeben sind.

Achten Sie darauf, dass Ihr **Name**, Ihre **Gruppe** und Ihre **Matrikelnummer** auf **jedem** Blatt stehen!

Falls Ihre Lösung mehrere Blätter umfasst, heften Sie diese bitte zusammen.

Aufgabe 1 – 18 Punkte

Klammeraffe

Überzeugen Sie sich, dass `Evaluate` (siehe `http://www.stud.informatik.uni-goettingen.de/info1/java/Evaluate.java`) tatsächlich wie am Ende von Kapitel 5 der Vorlesung behauptet, vollständig geklammerte Ausdrücke sowohl in Infix-Schreibweise, z.B.

```
( 1 + ( ( 2 + 3 ) * ( 4 * 5 ) ) )
```

als auch in Postfix-Schreibweise, z.B.

```
( 1 ( ( 2 3 + ) ( 4 5 * ) * ) + )
```

richtig verarbeitet.

1. Betrachten Sie folgende Ausdrücke in Infix-Schreibweise a), b) und in Postfix-Schreibweise c), d).

   a) `( 1 + ( ( 2 + 3 ) * ( 4 * 5 ) ) )`

   b) `1 + 2 + 3 * 4 * 5`

   c) `( 1 ( ( 2 3 + ) ( 4 5 * ) * ) + )`

   d) `1 2 3 + 4 5 * * +`

   Fügen Sie den Ausdrücken b) und d) jeweils möglichst wenig Zeichen hinzu, sodass diese genauso ausgewertet werden wie die Ausdrücke a) und c).
   (8 Punkte)

   Hinweis. Achten Sie auf die Funktion der Klammern bei dieser Implementierung.

2. Betrachten Sie folgenden Version von `Evaluate` mit Zeilennummern.

```
1  public class Evaluate {
2      public static void main(String[] args) {
3          Stack<String> ops  = new EVLStack<String>();
4          Stack<Double> vals = new EVLStack<Double>();
5
6          while (!StdIn.isEmpty()){          // Strings ..
7              String s = StdIn.readString(); // .. whitespace-getrennt
8              if      (s.equals("("))               ;
9              else if (s.equals("+"))    ops.push(s);
10             else if (s.equals("*"))    ops.push(s);
11             // usw.: alle Operatoren
12             else if (s.equals("-"))    ops.push(s);
13             else if (s.equals("/"))    ops.push(s);
14             else if (s.equals("sqrt")) ops.push(s);
15             else if (s.equals(")")) {
16                 String op = ops.pop();
17                 double v = vals.pop();
18                 if      (op.equals("+"))    v = vals.pop() + v;
19                 else if (op.equals("*"))    v = vals.pop() * v;
20                 // usw.: alle Operatoren
21                 else if (op.equals("-"))    v = vals.pop() - v;
22                 else if (op.equals("/"))    v = vals.pop() / v;
23                 else if (op.equals("sqrt")) v = Math.sqrt(v);
24                 vals.push(v);
25             }
26             else vals.push(Double.parseDouble(s));
27         }
28         StdOut.println(vals.pop());
29     }
30 }
```

Wie muss die Klasse modifiziert werden, sodass nicht geklammerte **und** vollständig geklammerte Ausdrücke in Postfix-Schreibweise korrekt verarbeitet werden, dabei müssen Ausdrücke in Infix-Schreibweise nicht mehr korrekt ausgewertet werden?

Beschreiben Sie die Modifikationen ausschließlich mit folgenden Operationen.

- Streichen von Zeile X
- Verschieben von Zeile X vor/hinter Zeile Y
- Ersetzen von Zeile X durch `... Java-Quelltext ...`
- Einfügen von `... Java-Quelltext ...` vor/nach Zeile X

Bemerkung

Einfügen, Streichen, Ersetzen von Zeilen ändert die Zeilennummer der ursprünglichen Zeilen nicht. Eingefügte Zeilen haben keine Zeilennummern, gestrichen Zeilen werden durch Leerzeilen ersetzt.

Insgesamt sind weniger als 8 Modifikationen notwendig, beginnen Sie wie folgt.

a) Verschieben von Zeile 26 hinter Zeile 14

b) ...

(10 Punkte)

Praktische Übung

Abgabe der Prüfsumme bis Fr., 25.01.2019, 23:55 Uhr.
Testat von Mo., 28.01.2019., 8-10 Uhr bis Fr., 01.02.2019, 18-20 Uhr.

Hilfe zum Bearbeiten der praktischen Übungen können Sie grundsätzlich jeden Tag in den Rechnerübungen bekommen, insbesondere in der Rechnerübung am **Montag, 21.01.2019**, in der **keine Testate** stattfinden.

Hinweise zu den praktischen Übungen, insbesondere zur **Abgabe der Prüfsumme** und zur **Durchführung der Testate**, sind in der Stud.IP-Veranstaltung *Informatik I* unter *Dateien→Übungsblätter* hinterlegt.

Aufgabe 1 – 45 Punkte

Schnittstellen, Vererbung, `Comparable<T>`

Eine Schnittstelle `Separable` für Zahlenformate, die in einen ganze Anteile `units` und gebroche Anteile `fractions` unterteilt werden können, ist wie folgt gegeben.

```
interface Separable {
  int getUnits();
  int getFractions();
  void setSeparator(char separator);
  void normalize();
}
```

Die `get`-Methoden sollen die gespeicherte Anzahl von `units` und `fractions` liefern. Mit `setSeparator` kann das Trennzeichen zwischen `units` und `fractions` für eine `String`-Repräsentation gesetzt werden. Die Methode `normalize` sorgt dafür, dass alle `fractions` in `units` überführt werden, bei denen das möglich ist.

Beispiel
Eurobeträge werden in Euros (`units`) und Cents (`fractions`) angegeben, das Trennzeichen ist ein Komma (`,`) und 100 Cents ergeben einen Euro, d.h. 1 Euro und 105 Cent würden von `normalize` auf 2 Euro und 5 Cent normalisiert werden. Die zugehörige `String`-Repräsentation ist `2,05 EUR`.

1. Erstellen Sie einen Klasse `Currency` für Währungsbeträge, die die Schnittstelle `Separable` implementiert, mit folgenden Eigenschaften.
   a) Alle Klassen- und Objektvariablen sind `private`.
   b) Alle Methoden aus `Separable` werden implementiert. Weil für nahezu alle Währungen der Welt gilt, dass 100 `fractions` einem `unit` entsprechen, können Sie das für `normalize` auch annehmen. Berücksichtigen Sie in `normalize` auch, dass nur für `units` negative Werte vorkommen dürfen, negative Werte für `fractions` müssen in nicht-negative Werte überführt werden.
   c) Die von `Object` geerbte Methode `toString` wird überschrieben, sodass eine sinnvolle `String`-Repräsentation aus `units`, `fractions` und Trennzeichen zurückgeliefert wird.

d) Erstellen Sie einen Default-Konstruktor, der ein Objekt erzeugt, dass einen `unit` repräsentiert.

e) Erstellen Sie einen Konstruktor, der `units` und `fractions`, sowie einen Konstruktor der `units`, `fractions` und Trennzeichen übergeben bekommt. Akzeptieren Sie alle möglichen Eingaben und benutzen Sie `normalize` um sicherzustellen, dass die Konstruktoren ein normalisiertes Objekt erzeugen.

f) Erstellen Sie einen Konstruktor, der ein `Currency`-Object übergeben bekommt und mit Hilfe eines Konstruktors aus Aufgabenteil 1e) eine Kopie davon anfertigt.

g) Sollten Sie zusätzliche Methoden benötigen, implementieren Sie diese nach Bedarf `protected` oder `private`.

2. Erstellen Sie einen Klasse `Euro` für Euro/Cent-Beträge, die von der Klasse `Currency` erbt, mit folgenden Eigenschaften.

   a) Die von `Currency` geerbte Methode `toString` wird so überschrieben, dass die übliche `String`-Repräsentation mit den hinterlegten Trennzeichen und `EUR` als Währungskennzeichnung geliefert wird, z.B. `2,05 EUR`.

   b) Implementieren Sie Methoden `getEuro` und `getCent`.

   c) Erstellen Sie eine Default-Konstruktor analog zu dem in `Currency`. Setzen Sie als Trennzeichen ein Komma (`,`).

   d) Erstellen Sie eine Konstruktor für `euro`- und `cent`-Werte. Setzen Sie das Trennzeichen.

3. Erstellen Sie einen Klasse `Ariary` für Ariary/Iraimbilanja-Beträge (Währung von Madagaskar), die von der Klasse `Currency` erbt, mit folgenden Eigenschaften.

   a) Die von `Currency` geerbte Methode `toString` wird so überschrieben, dass einen beschreibende `String`-Repräsentation mit `MAG` als Kennzeichnung der `units` und Iraimbilanja als Kennzeichnung der `fractions` geliefert wird, z.B. `5 MAG, 3 Iraimbilanja`.

   b) Implementieren Sie Methoden `getAriary` und `getIraimbilanja`.

   c) Erstellen Sie eine Default-Konstruktor analog zu dem in `Currency`.

   d) Erstellen Sie eine Konstruktor für `ariary`- und `iraimbilanja`-Werte.

   e) Für die Währung von Madagaskar gilt, dass 5 Iraimbilanja einem Ariary entsprechen. Überschreiben Sie die Methode `normalize` so, dass dieser Zusammenhang berücksichtigt wird.

4. Erstellen Sie eine Klasse `Dollar` für Dollar/Cent-Beträge, die von der Klasse `Currency` erbt, analog zur Klasse `Euro`. Erweitern Sie `Dollar` um die Möglichkeit andere Währungen in Dollar umzutauschen wie folgt.

   a) Implementieren Sie eine Hilfsmethode `exchange` als `private` Klassenmethoden, die ein neues `Dollar`-Object zurück liefert, dessen Wert aus einem übergebene `Currency`-Object, Faktor (Anzahl `fractions` pro `unit`) und Wechselkurs berechnet wird.

Überladen Sie `exchange` mit einer `public` Klassenmethoden, die ein `Euro`-Object übergeben bekommt und ein neues `Dollar`-Object zurück liefert, dazu wird die Hilfsmethode verwendet.

Überladen Sie `exchange` mit einer `public` Klassenmethoden, die ein `Ariary`-Object übergeben bekommt und ein neues `Dollar`-Object zurück liefert, dazu wird die Hilfsmethode verwendet.

b) Hinterlegen Sie die Wechselkurse als Klassenkonstanten in `Dollar` wie folgt.

```
public static final double EURO_RATE   = 0.8152;
public static final double ARIARY_RATE = 3245.0;
```

5. Erweitern Sie die Klasse `Dollar`, sodass die generische Schnittstelle `Comparable<T>` implementiert wird.

```
public class Dollar implements Comparable<Dollar>
```

Implementieren Sie die Methode `compareTo` wie dort gefordert, mit Ausnahme der geforderten Exception (Abschnitt *Throws*).

Bemerkung
Informieren Sie sich über die generische Schnittstelle `Comparable` im Java-API `https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html` .

6. Erstellen Sie ein Test-Unit, dass alle Klassen ausgiebig testet.

7. Versehen Sie Ihren Code mit ausführlichen Kommentaren.

(45 Punkte)