Praktikum Grundlagen der Programmierung

Prof. Dr. Harald Räcke, R. Palenta, A. Reuss, S. Schulze Frielinghaus WS 2016/17

Übungsblatt 4 Abgabefrist: 21.11.2016 (vor 05:00 Uhr)

Um Ihre Hausaufgaben auf Moodle abgeben zu können, müssen Sie in einer Gruppe in TUMonline eingetragen sein!

Hausaufgaben, die keine Programieraufgaben sind, müssen als PDF im A4-Format abgegeben werden.

Aufgabe 4.1 (P) Java-Quiz

	Wahr	Falsch

Folgende Programmstücke sind semantisch äquivalent
`int x = read(); int y = x/x;`
und
`int x = read(); int y = 1;`
	☐	☒

Lösung: Die Methode `read()` könnte die Zahl 0 zurück liefern und somit würde im ersten Programmstück eine `ArithmeticException` (division by zero) geworfen, was im zweiten Fall nicht passieren kann. Somit sind die Programme nicht semantisch äquivalent.

Der Ausdruck `0.3 == 0.1 + 0.1 + 0.1` evaluiert zu `true`	☐	☒

Lösung: Mathematisch betrachtet sollte der Ausdruck wahr sein. Die Gleitkommazahl 0.1 kann in Java aber nicht exakt dargestellt werden. D.h. es wird nur mit einer Approximation/Annhäherung der Zahl 0.1 gerechnet. Somit kann das Ergebnis aber auch nicht exakt sein. Als Lehre nehmen wir mit, dass Gleitkommazahlen im Generellen nicht auf Gleichheit überprüft werden sollten, sondern ob diese in einem *Wertebreich* liegen.

Mit dem 32-bit Datentyp `float` lassen sich größere Zahlen darstellen als mit dem 64-bit Datentyp `long`	☒	☐

Lösung: Obwohl für den Datentyp `float` weniger Bits zur Verfügung stehen als in einem `long`, können damit größere Zahlen dargestellt werden. Das liegt an der Darstellung von Gleitkommazahlen. Diese werden mittels Vorzeichen, Mantisse und Exponent dargestellt. Durch die Interpretation einiger Bits als Exponenten, lassen sich somit große Zahlen darstellen.

Jeder 32-bit `Integer` kann in einem 32-bit `Float` dargestellt werden.	☐	☒

Lösung: Try it out with the `Integer` `16777217`

Der Ausdruck `(x + y) - y == x` evaluiert immer zu `true` unter der Annahme, dass `x` und `y` vom selben Zahlentyp sind.	☐	☒

Wahr Falsch

Lösung: Für Gleitkommazahlen gilt dies nicht. Wenn `y` den Wert `NaN` oder `POSITIVE_INFINITY` bzw. `NEGATIVE_INFINITY` hat muss die Gleichheit nicht gelten. Ein weiteres Argument ist ähnlich zu der Aufgabe weiter oben. Versuchen Sie es mit den Werten `0.3` für `x` und `0.1` für `y`.

Angenommen `x` und `y` sind vom Typen `int`, dann sind folgende Ausdrücke semantisch äquivalent
`x + 10 > y + 10`
und
`x > y`

☐ ☒

Lösung: Angenommen, dass `x + 10` zu einem *Integer Overflow* führt und `y + 10` nicht, dann sind die Ausdrücke nicht äquivalent. Beispiel: `x = Integer.MAX_VALUE - 1; y = 0;` **Achtung:** Treten zwei Integer Overflows auf, dann sind die Ausdrücke wieder äquivalent!

Der Ausdruck `2 + 5 + ">=" + 1 + 1` evaluiert zu `"7>=11"`

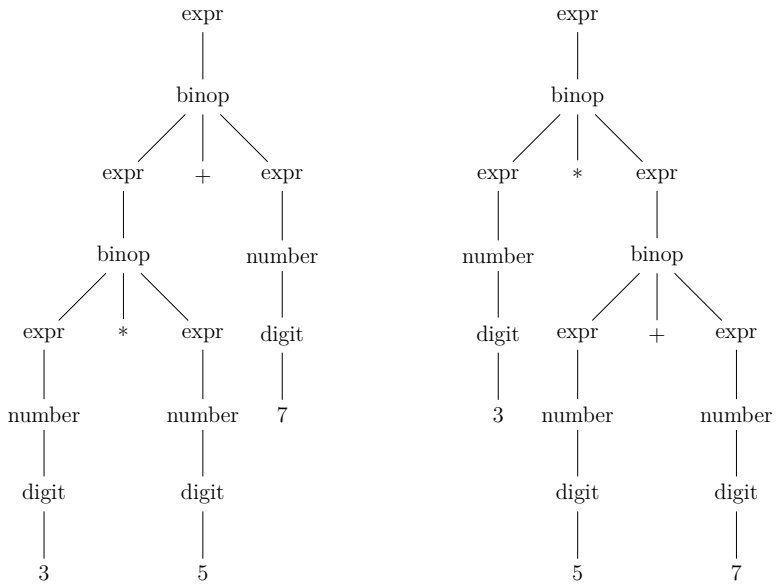
☒ ☐

Lösung: *Operator Overloading* is evil! Machen Sie sich mit der Auswertereihenfolge von Ausdrücken bekannt: `((((2 + 5) + ">=") + 1) + 1)`. Vereinfacht gesagt wird bei Java wie folgt ein Additionsausdruck ausgewertet: Zahl + Zahl ergibt eine Zahl, Zahl + String oder String + Zahl ergibt einen String.

Gegeben ist folgender Ausdruck `3*5+7`. Ist der Syntaxbaum/Ableitungsbaum, der mithilfe der MiniJava-Grammatik erzeugt werden kann, eindeutig?

☐ ☒

Lösung: Kurze Antwort: Nein, es gibt zwei Ableitungsbäume:



Wahr Falsch

Lange Antwort: Für das Wort `3*5+7` existieren zwei unterschiedliche Linksableitungen. Somit können wir schließen, dass die MiniJava-Grammatik mehrdeutig ist :(

Aufgabe 4.2 (P) Vokalersetzung

Schreiben Sie ein Programm `Vokalersetzung`, das im unten gegebenen Text alle Vokale durch einen vom Benutzer eingegebenen Vokal ersetzt. Achten Sie darauf, dass die Groß- und Kleinschreibung nach der Ersetzung mit der vor der Ersetzung übereinstimmt. Umlaute (Ä/ä, Ö/ö, usw.) sollen dabei nicht als Vokale betrachtet werden. Beispiel: Das Wort `"Exenmeister"` wird bei Eingabe von O oder o zu `"Oxonmoostor"`.

Verwenden Sie keine Bibliotheksfunktionen der Klasse `String` außer `char charAt(int i)` und `int length()`.

- `int length()`
  Um die Länge eines Strings zu bestimmen rufen Sie die Methode `length()` auf.

  Beispiel: `String s = "Hello Students"; int l = s.length();` speichert den Wert `14` in der Variable `l`.

- `char charAt(int index)`
  Um auf ein Zeichen innerhalb eines Strings zuzugreifen, können sie die Methode `charAt(i)` verwenden, welche ein `Character` an der Position `i` zurück liefert.

  Beispiel: `String s = "Hello Students"; char c = s.charAt(7);` speichert den `Character 't'` in der Variable `c`. Achtung: Java indiziert ab der Stelle 0 und nicht erst ab 1.

Hinweis: Eine `Integer`-Expression `e` können Sie wie folgt in einen Character umwandeln: `(char) e`. Zum Beispiel `int i = 64; char c = (char) i;` speichert in der `Character`-Variable `c` das Zeichen `'A'`. Machen Sie sich mit der ASCII-Tabelle vertraut.

Verwenden Sie das folgende Codegerüst:

```
public class Vokalersetzung extends MiniJava {

    public static void main(String[] args) {

        String text = "Hat der alte Hexenmeister\n" +
                      "sich doch einmal wegbegeben!\n" +
                      "Und nun sollen seine Geister\n" +
                      "auch nach meinem Willen leben.\n" +
                      "Seine Wort und Werke\n" +
                      "merkt ich und den Brauch,\n" +
                      "und mit Geistesstärke\n" +
                      "tu ich Wunder auch.\n" +
                      "Walle! walle\n" +
                      "Manche Strecke,\n" +
                      "daß, zum Zwecke,\n" +
                      "Wasser fließe\n" +
                      "und mit reichem, vollem Schwalle\n" +
```

```
                    "zu dem Bade sich ergieße.";
    }

}
```

Aufgabe 4.3 (P) Rechtschreibschwäche

Schreiben Sie ein MiniJava-Programm, dass einen `String` einliest und jeden Kleinbuchstaben durch den jeweiligen Grossbuchstaben und umgekehrt ersetzt. Alle anderen Zeichen sollen nicht ersetzt werden. Geben Sie den konvertierten `String` via der Methode `write(String s)` aus.

Beispiel: `"Hello Students!"` wird umgewandelt in `"hELLO sTUDENTS!"`.

Folgende Methoden können Sie zusätzlich verwenden:

- `int length()`
  Um die Länge eines Strings zu bestimmen rufen Sie die Methode `length()` auf.

  Beispiel: `String s = "Hello Students"; int l = s.length();` speichert den Wert `14` in der Variable `l`.

- `char charAt(int index)`
  Um auf ein Zeichen innerhalb eines Strings zuzugreifen, können sie die Methode `charAt(i)` verwenden, welche ein `Character` an der Position `i` zurück liefert.

  Beispiel: `String s = "Hello Students"; char c = s.charAt(7);` speichert den `Character 't'` in der Variable `c`. Achtung: Java indiziert ab der Stelle 0 und nicht erst ab 1.

Hinweis: Eine `Integer`-Expression `e` können Sie wie folgt in einen Character umwandeln: `(char) e`. Zum Beispiel `int i = 65; char c = (char) i;` speichert in der `Character`-Variable `c` das Zeichen `'A'`. Machen Sie sich mit der ASCII-Tabelle vertraut.

Verwenden Sie nur die Methoden aus dem Angabentext oder welche MiniJava bereit stellt.

Aufgabe 4.4 (P) Funktionsaufrufe

Gegeben sind die folgenden Funktionssignaturen

```
double max(double a, int b)
double max(int a, double b)
double max(double a, float b)
```

- Vergleichen Sie alle Funktionen miteinander. Welche ist spezifischer als die andere? Welche sind unvergleichbar?
- Betrachten Sie folgenden Funktionsaufruf `max(3, x)`. Welche primitiven numerischen Typen kann die Variable `x` haben, so dass der Aufruf mit den oben gegebenen Funktionen nicht ungültig ist?
- Welche der folgenden Funktionsaufrufe sind gültig? Welche der Funktionen wird bei einem gültigen Aufruf verwendet?

```
max(3.0, 5.0);
max(3f, 3);
max(3f, 3.0);
max(3.0, 4f);
max(5.0, 2);
```

Lösungsvorschlag 4.4

- – `double max(double a, int b)` und `double max(int a, double b)` sind unvergleichbar
  – `double max(int a, double b)` und `double max(double a, float b)` sind unvergleichbar
  – `double max(double a, int b)` ist spezifischer als `double max(double a, float b)`
- Die Variable `x` kann lediglich den Typen `double` haben, da bei allen anderen Typen immer zwei unvergleichbare Funktionen zur Wahl stehen.
- ```
  max(3.0, 5.0);        /* ungueltiger Aufruf,
                           da es keine passende Funktion gibt */
  max(3f, 3);             // ruft max(double a, int b) auf
  max(3f, 3.0);             /* ungueltiger Aufruf,
                           da es keine passende Funktion gibt */
  max(3.0, 4f);             // ruft max(double a, float b) auf
  max(5.0, 2);             // ruft max(double a, int b) auf
  ```

Aufgabe 4.5 (H) Seiteneffekte [7.5 Punkte]

Geben Sie für jeden Ausdruck eine Folge von Ausdrücken an, so dass die Variablen im Ausdruck nach der Auswertung den gleichen Wert haben wie nach der Auswertung, der von Ihnen angegeben Ausdrücke.

In den Ausdrücken dürfen Sie nur Wertzuweisungen, Addition und Subtraktion (keine Inkrement-/Dekrementoperatoren) verwenden. Sie dürfen keine zusätzlichen Variablen verwenden. Bei allen Variablen handelt es sich um `Integer`.

1. `c = b++ + ++b - --b;`
2. `a = d + 1 - --d;`
3. `e = f++ + f++ + f++;`

Lösungsvorschlag 4.5

1. ```
   c = b;
   b = b + 1;
   b = b + 1;
   c = c + b;
   b = b - 1;
   c = c - b;
   ```

2.
```
a = d + 1;
d = d - 1;
a = a - d;
```

3.
```
e = f;
f = f + 1;
e = e + f;
f = f + 1;
e = e + f;
f = f + 1;
```

Korrekturbemerkung:

- pro Anweisung 0.5 Punkte
- Zusammenfassungen wie `b = b + 2;` sind erlaubt und ergeben entsprechend mehr Punkte
- prinzipiell reichen zwei Ausdrücke je Teilaufgabe
- Punkte auf Zwischenschritte aber falschem Endergebnis kann es nur geben, wenn diese erkennbar sind

Aufgabe 4.6 (H) Cäsar der Kryptische [6 Punkte]

Schreiben Sie ein Programm, dass einen `String` einliest und diesen mittels der Cäsar-Chiffre verschlüsselt. Machen Sie sich mit der Cäsar-Verschlüsselung vertraut. Dazu ist die Wikipedia Ihr Freund und Helfer:
`https://de.wikipedia.org/wiki/Caesar-Verschl%C3%BCsselung`

Der Einfachheit halber werden wir nur Buchstaben `a` bis `z` und `A` bis `Z` verschlüsseln. D.h. alle anderen Zeichen sollen nicht verschlüsselt werden und tauchen somit als Klartext im Geheimtext auf.

Beispiel: Der Geheimtext zum Klartext `"Hello Students! .aAbBcC? >wWxXyYzZ<"` lautet `"Khoor Vwxghqwv! .dDeEfF? >zZaAbBcC<"` wenn der zyklische Shift 3 beträgt.

Gehen Sie wie folgt vor:

(a) Lesen Sie einen `String` mittels der MiniJava-Methode `readString()` ein, der später ver/entschlüsselt werden soll.

(b) Lesen Sie einen `Integer` mittels der MiniJava-Methode `read()` ein, der als zyklischer Shift verwendet werden soll. Beachten Sie, dass der `Integer` sowohl negativ als auch betragsmäßig größer als 26 sein kann!

(c) Implementieren Sie nun den Algorithmus und ver/entschlüsseln den `String` aus (a) anhand des Schlüssels aus (b).

(d) Beachten sie Groß- und Kleinschreibung im Klartext als auch Geheimtext.

(e) Geben Sie den verschlüsselten `String` mittels der MiniJava-Methode `write(String s)` aus.

Folgende Methoden können Sie zusätzlich verwenden:

- `int length()`
  Um die Länge eines Strings zu bestimmen rufen Sie die Methode `length()` auf.

  Beispiel: `String s = "Hello Students"; int l = s.length();` speichert den Wert `14` in der Variable `l`.

- `char charAt(int index)`
  Um auf ein Zeichen innerhalb eines Strings zuzugreifen, können sie die Methode `charAt(i)` verwenden, welche ein `Character` an der Position `i` zurück liefert.

  Beispiel: `String s = "Hello Students"; char c = s.charAt(7);` speichert den `Character 't'` in der Variable `c`. Achtung: Java indiziert ab der Stelle 0 und nicht erst ab 1.

Hinweis: Sie können Ihre Implementierung selber testen, indem Sie den Geheimtext mit dem Schlüssel `-n` nochmals "verschlüsseln", sofern der Geheimtext vorher mit dem Schlüssel `n` verschlüsselt wurde. Die Ausgabe sollte der initiale Klartext sein. Erinnert Sie das an inverse Funktionen von bijektiven Funktionen à la $x = f^{-1}(f(x))$?

Verwenden Sie nur die Methoden aus dem Angabentext oder welche MiniJava bereit stellt.

Lösungsvorschlag 4.6

Korrekturbemerkung:

- Es dürfen nur Zeichen `a` bis `z` und `A` bis `Z` verschlüsselt werden. Alle anderen Zeichen tauchen als Klartext im Geheimtext auf: 2 Punkte
- Gross- und Kleinschreibung im Geheimtext entspricht der Gross- und Kleinschreibung des Klartexts: 1 Punkt
- Zyklische Verschiebung beachtet: 3 Punkte

Aufgabe 4.7 (H) Verhext [8 Punkte + 2 Bonuspunkte]

Schreiben Sie ein MiniJava-Programm welches eine Zahl in hexadezimaler Form als `String` einliest und diese in einen `Integer` umwandelt. Verwenden Sie dafür folgendes Grundgerüst:

```
public class Verhext extends MiniJava {

    // x^y
    public static int pow(int x, int y) {
        return java.math.BigInteger.valueOf(x).pow(y).intValueExact();
    }

    public static void main(String[] args) {
        String input = readString();
        int output;

        /* input your solution here */

        write(output);
    }
}
```

Sie können davon ausgehen, dass der Eingabestring eine gültige Zahl in hexadezimaler Form ist. Desweiteren können Sie davon ausgehen, dass der String eine Zahl beschreibt, die in einen `Integer` passt, d.h. ein Suffix `l` oder `L` kommt nicht vor.

Sie können wie folgt vorgehen:

- Schreiben Sie eine erste Implementierung, die auf den Basisfällen wie z. B. `"0x5Ab"`, `"0Xabc"`, `"0X1A3"` funktioniert.
- Nun erweitern Sie Ihr Programm, sodass auch Eingaben mit Unterstrichen funktionieren: `"0xA_B_C"`, `"0X98_F_3"`
- Zu guter Letzt sollen auch negative Zahlen eingelesen werden können: `"-0xFfF"`
- Bonus-Aufgabe (2 Punkte): Fangen Sie fehlerbehaftete Eingaben ab wie z. B. `"0xX"`, `"--0xABC"` und vieles vieles mehr. Be creative ;-)

Folgende Methoden können Sie zusätzlich verwenden:

- `int length()`
  Um die Länge eines Strings zu bestimmen rufen Sie die Methode `length()` auf.
  Beispiel: `String s = "Hello Students"; int l = s.length();` speichert den Wert `14` in der Variable `l`.
- `char charAt(int index)`
  Um auf ein Zeichen innerhalb eines Strings zuzugreifen, können sie die Methode `charAt(i)` verwenden, welche ein `Character` an der Position `i` zurück liefert.
  Beispiel: `String s = "Hello Students"; char c = s.charAt(7);` speichert den `Character 't'` in der Variable `c`. Achtung: Java indiziert ab der Stelle 0 und nicht erst ab 1.
- `int pow(int x, int y)`
  Die Potenz eines `Integers` können Sie mit der Methode `pow` berechnen.
  Beispiel: `pow(3,5)` liefert als Wert `243`.

Verwenden Sie nur die Methoden aus dem Angabentext oder welche MiniJava bereit stellt.

Lösungsvorschlag 4.7

Korrekturbemerkung:

- Basisfälle: 3 Punkte
- Unterstrich: 2 Punkte
- Negative Zahlen: 3 Punkte
- Bonus-Aufgabe: 2 Punkte