HUMBOLDT-UNIVERSITÄT ZU BERLIN

Hausaufgaben zur Veranstaltung

Einführung in die Programmierung in Java

- 2. Blatt -

Abgabe bis 27.11.17 um 9:00 Uhr

Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Σ
3	4	4	4	5	20

Die Hausaufgaben werden in den Übungen am 28.11. und 29.11. besprochen. Die Abgabe der Lösungen erfolgt ausschließlich durch Hochladen der Lösungen bzw. der Java-Programme [`Programmname.java`] in Moodle. Programme, die nicht kompiliert werden können (weil sie z. B. syntaktische Fehler enthalten), werden mit **0 Punkten** bewertet.

1. Aufgabe (1+1+1 Punkte)

In Java werden Operationen in einer bestimmten Reihenfolge ausgewertet. Dabei haben die Operatoren unterschiedliche Priorität (Präzedenz). Die Liste auf der folgenden Seite stellt einen Auszug[1] der in Java möglichen Operationen und ihrer Priorität dar. Je höher der Operator in der Liste steht, desto höher ist die Priorität des Operators. Bei gleicher Priorität hängt die Auswertung eines Ausdruck von der Assoziativität ab. Entweder erfolgt die Auswertung

von links nach rechts: `a - b + c - d` ↔ `((a - b) + c) - d`

oder

von rechts nach links: `a += b = c -= d` ↔ `a += (b = (c -= d))`.

[1] Eine vollständige Übersicht finden Sie unter https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html.

Operatoren	**Assoziativität**
`expr++ expr--` `(Post-Inkrement/-Dekrement)`	`links nach rechts`
`++expr --expr +expr -expr` `(Prä-Inkrement/-Dekrement und Vorzeichenänderung)`	`rechts nach links`
`* / %`	`links nach rechts`
`+ -`	`links nach rechts`
`< > <= >=`	`links nach rechts`
`== !=`	`links nach rechts`
`&&`	`links nach rechts`
`||`	`links nach rechts`
`= += -= *= /= %=`	`rechts nach links`

Klammern Sie die folgenden Ausdrücke so, wie sie ausgewertet werden und bestimmen Sie, von welchem Typ (`Integer` oder `Boolean`) jeweils die Variable `result` sein muss. Die auftretenden Variablen `a, b, c, ... , f` sind vom Typ `Integer` oder vom Typ `Boolean`, die sich jeweils aus dem Kontext der verwendeten Operation ergeben. Ein Ausdruck kann dabei Variablen unterschiedlichen Typs enthalten.

Der Ausdruck `result = ++a * b + c / d` hätte z. B. die Klammerung `((++a) * b) + (c / d)` und die Variable `result` müsste vom Typ `Integer` sein.

a) `result = a * b + ++c - d >= e - f`

☐ `Integer` ☐ `Boolean`

b) `result = a == b && c || d < e / -f`

☐ `Integer` ☐ `Boolean`

c) `result = a * b * c + d - e-- / f`

☐ Integer ☐ Boolean

2. Aufgabe (2+2 Punkte)

Betrachten Sie den folgenden Programmausschnitt:

```
int n = Integer.parseInt(args[0]); // initial hat n den Wert 7
long s = 0;
for (int i = 1, j = n; i < n; i += 3, j -= 2) {
      s += i * j;
}
s *= 10;
```

a) Vervollständigen Sie die folgende Variablenbelegungstabelle für `n = 7`. Ergänzen Sie eine Zeile, wenn der Wert einer der angegebenen Variablen verändert wird. In jeder Tabellenzeile darf nur **eine** Variable verändert werden.

n	**s**	**i**	**j**	**Anmerkung** **(Spalte muss nicht ausgefüllt werden!)**
-	-	-	-	Start der Ausführung
7	-	-	-	Übergabe Kommandozeilenparameter als Integer an die Variable **n**.
7	0	-	-	Deklaration und Initialisierung der Variable **s** mit dem Wert 0.

b) Schreiben Sie den Programmausschnitt so um, dass statt einer for- eine while-Schleife verwendet wird. Die mathematische Operation `s += i * j` im Schleifenkörper darf nicht verändert werden.

```
// hier könnte Code stehen...
```

3. Aufgabe [`Numbers.java`] (4 Punkte)

Erweitern Sie das Programm `Numbers` so, dass ein Array mit 50 Einträgen vom Typ Integer erstellt und dieses anschließend mithilfe **von (einer) Schleife(n)** nach folgendem Muster mit Werten gefüllt wird:

Die Einträge in der ersten Hälfte (Index 0 bis 24) sollen aufsteigend mit geraden Zahlen von 1 bis 50 gefüllt werden. Die Einträge in der zweiten Hälfte (Index 25 bis 49) sollen absteigend mit ungeraden Zahlen von 50 bis 1 gefüllt werden. Anschließend soll das Array auf der Konsole ausgegeben werden, wobei alle Einträge durch ein Komma separiert in einer Zeile ausgegeben werden sollen. Die Ausgabe soll folgendermaßen aussehen:

```
2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34,
36, 38, 40, 42, 44, 46, 48, 50, 49, 47, 45, 43, 41, 39, 37, 35,
33, 31, 29, 27, 25, 23, 21, 19, 17, 15, 13, 11, 9, 7, 5, 3, 1
```

Hinweis: Aufgrund der begrenzten Anzahl an Zeichen, die pro Zeile in der Konsole angezeigt werden können, kann die Ausgabe über mehrere Zeilen laufen. Das wird nicht als Fehler bemängelt.

4. Aufgabe [`Duplicates.java`] (4 Punkte)

Erweitern Sie das Java-Programm `Duplicates` um eine statische Methode mit dem Namen `find`, die ein Array mit Werten vom Typ `String` als Parameter besitzt und einen Wert vom Typ `Boolean` zurückgibt. Die Methode soll entscheiden, ob in dem Array Werte mehrfach vorhanden sind. Das Programm soll mit einer unbestimmten, aber nicht leeren Menge von Kommandozeilenparametern aufgerufen und die Menge soll mithilfe der Methode `find` überprüft werden. Das Ergebnis der Überprüfung soll auf der Konsole ausgegeben werden.

```
% java Duplicates Jürgen Michael Wolfgang Heidi Michael Rüdiger
In der Menge sind Werte mehrfach vorhanden.

% java Duplicates Heidi Simone Steffen Rüdiger Otto Stefan Susi
In der Menge sind keine Werte mehrfach vorhanden.
```

Hinweise: Die Kommandozeilenparameter sollen nicht geparst, sondern als Zeichenketten behandelt werden. Sie können davon ausgehen, dass dem

Programm immer mindestens 2 Kommandozeilenparameter übergeben werden. Die Anzahl der Parameter ist nach oben jedoch nicht begrenzt.

5. Aufgabe [`LoveCalculator.java`] (5 Punkte)

Erweitern Sie das Programm `LoveCalculator` so, dass für zwei Namen getestet wird, wie gut die beiden Personen zusammenpassen.

Schreiben Sie zunächst eine statische Methode mit dem Namen `sum`, die für eine Zeichenkette die Summe aller Buchstaben berechnet und als Integer zurückgibt. Wenden Sie die Methode jeweils auf den ersten beiden Kommandozeilenparameter an. Übergeben Sie anschließend die beiden errechneten Ergebnisse der Methode `calculate`, die eine Berechnung durchführt und das Ergebnis als Integer zurückgibt. Das Ergebnis der Methode `calculate` liegt zwischen 1 und 100 und sagt aus, wie gut die beiden Personen zusammenpassen. Es sollen folgende Abstufungen gelten:

Ergebnis zwischen 1 und 20: Bei euch beiden gefriert die Hölle.
Ergebnis zwischen 21 und 40: Ihr seid wie Hund und Katze.
Ergebnis zwischen 41 und 60: Da ist Langeweile vorprogrammiert.
Ergebnis zwischen 61 und 80: Bei euch beiden sprühen die Funken.
Ergebnis zwischen 81 und 100: Das ist wahre Liebe.

Geben Sie anschließend abhängig vom finalen Ergebnis eine Zeichenkette ("Bei euch beiden gefriert die Hölle.", …, "Das ist wahre Liebe.") aus.

```
% java LoveCalculator Susi Gustav
Ihr seid wie Hund und Katze.

% java LoveCalculator Hans Gertrud
Bei euch sprühen die Funken.
```

Hinweise: Die Methode `calculate` darf nicht verändert werden. Um den Wert eines Buchstabens zu erhalten, verwenden Sie folgende Funktion aus der Klasse Character: `public static int getNumericValue(char c)`

```
char c1 = 'A', c2 = 'T';
int value1 = Character.getNumericValue(c1); // value1 hat den Wert 10
int value2 = Character.getNumericValue(c2); // value2 hat den Wert 29
```