Praktikum Grundlagen der Programmierung

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

Übungsblatt 2 Abgabefrist: 07.11.2016 (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.

Entfernen Sie aus allen Dateien, die Sie als Hausaufgabe im Praktikum abgeben, Ihre personenbezogenen Daten.

Aufgabe 2.1 (P) Grammatik Integer

Ab Java SE 7 können beliebig viele Unterstriche zwischen Ziffern in numerischen Literalen auftreten. Diese Unterstriche haben keine semantische Bedeutung und dienen lediglich der Lesbarkeit. Beispiel: `int x = 100_000;` wird interpretiert wie `int x = 100000;`.

In der Vorlesung wurde eine Grammatik für ganze Zahlen mit führenden Nullen angegeben. Geben Sie eine Grammatik für ganze Zahlen *ohne* führende Nullen und mit den ab Java SE 7 erlaubten Unterstrichen für numerische Literale an. Beachten Sie, dass die Unterstriche nur zwischen Ziffern, nicht aber am Anfang oder Ende eines numerischen Literals erlaubt sind, siehe `http://docs.oracle.com/javase/7/docs/technotes/guides/language/underscores-literals.html`.

Lösung 2.1

Die Lösung befindet sich in der Datei `Integer_Sol.txt`.

Aufgabe 2.2 (P) Reguläre Ausdrücke

Geben Sie unter Verwendung der in der Vorlesung für reguläre Ausdrücke eingeführten Operatoren *, ? und | einen regulären Ausdruck für $^+$ an. Wobei a^+ bedeutet, dass das Terminal a ein oder mehrmals auftreten darf.

Geben Sie reguläre Ausdrücke für die folgenden Textmuster, wobei `letter` einen beliebigen Buchstaben im Text beschreibt:

1. Der Text enthält kein b.
2. Der Text beginnt mit a oder b und endet mit c.
3. Der Text beginnt mit a und endet mit b oder der Text beginnt mit b und endet mit a.

Es gilt `letter` $::= a|\ldots|z$.

Lösung 2.2

Die Lösung befindet sich in der Datei `l.txt`.

Aufgabe 2.3 (P) Meiern

In dieser Aufgabe wollen wir das Würfelspiel „Meiern“ implementieren, das auf den folgenden Regeln basiert:

> Für das Spiel braucht man zwei Würfel. Ein Spieler tritt gegen den Computer an. In jeder Spielrunde würfelt der Spieler mit beiden Würfeln. Die beiden gewürfelten Zahlen kombiniert er zu einer zweistelligen Zahl, so dass die größere der beiden als 10er Stelle und die kleinere von beiden als 1er-Stelle interpretiert wird.
>
> Für die Rangfolge der Würfe gilt:
>
> Der höchste Wurf (21), der sogenannte „Meier“, ist durch keinen anderen Wurf zu überbieten. Ein Spieler, der einen „Meier“ würfelt, hat sofort gewonnen.
> Kurz nach dem „Meier“ folgt der *6er-Pasch* (66), gefolgt von den anderen Paschen in absteigender Reihenfolge. Auf den kleinsten Pasch (11) folgen dann die normalen Würfe in absteigender numerischer Reihenfolge von 65 bis hinab zum kleinstmöglichsten Wurf (31).
> Es wird so lange abwechselnd von Spieler und Computer gewürfelt, bis der aktuelle Spieler den Wurf des vorherigen Spielers nicht mehr überbietet und deshalb das Spiel verliert (Ausnahme ist der „Meier“ wie zuvor beschrieben).

Schreiben Sie ein `MiniJava`-Programm, mit dem man „Meiern“ gegen den Computer spielen kann. Der Spieler soll abwechselnd über Dialogboxen über seine und die Würfe des Computers informiert werden.

Realisieren Sie diese Aufgabe Schritt für Schritt:

- Die Klasse `MiniJava` stellt Ihnen eine Methode dice() zur Verfügung, die Ihnen einen zufälligen Zahlenwert von 1 bis 6 liefert.
- Um eine Zahl zwischen 1 und 6 zu würfeln, verwenden Sie beispielsweise den folgenden Code: `int zahl; zahl = dice();`.
- Werfen Sie zwei Würfel und bestimmen Sie den Wert des Wurfes.
- Behandeln Sie die besondere Rangfolge der Würfe.
- Erweitern Sie das Programm, so dass die Würfe von Spieler und Computer unterschieden werden können.
- Merken Sie sich den Wurf des vorherigen Spielers.
- Denken Sie daran, den Spieler über die Würfe und über Sieg oder Niederlage zu informieren.

Lösung 2.3

Die Lösung befindet sich in der Datei `Meiern.java`.

Aufgabe 2.4 (P) Lustige Sieben

In dieser Aufgabe wollen wir das Würfelspiel *Lustige Sieben* als ein Programm namens `LustigeSieben.java` schreiben. Es gibt einen Spieler, der gegen die Bank spielt. Der Spieler startet mit einem Guthaben von 100. Der Spieler setzt einen Teil seines Guthabens auf nur eines der Felder des Spielfelds. Die Bank würfelt mit zwei Würfeln. Dazu steht Ihnen die Methode `int dice()` der Klasse `MiniJava` zur Verfügung. Anschließend zahlt die Bank entsprechend folgender Regel an den Spieler dessen Gewinn aus:

- Der dreifache Einsatz wird ausgezahlt, falls die Summe der beiden Würfel 7 ergibt und der Spieler auf die 7 gesetzt hat;
- der doppelte Einsatz wird ausgezahlt, falls die Summe der beiden Würfel genau der gewählten Zahl des Spielfeldes entspricht;
- der einfache Einsatz wird zurückgezahlt, falls sich das Würfelergebnis auf derselben Längsseite wie die gewählte Zahl befindet.

Beispiel: Wenn insgesamt 4 Augen gewürfelt werden, so erhält der Spieler für die Wahl der 4 den doppelten Einsatz, seinen Einsatz zurück, wenn der Spieler auf die $2, 3, 5$ oder 6 gesetzt hat, und verliert andernfalls seinen Einsatz an die Bank.

Das Spielfeld sieht wie folgt aus:

7	
2	8
3	9
4	10
5	11
6	12

Das Programm fragt so lange nach der gewählten Zahl und dem Einsatz, bis das Guthaben von 100 des Spielers aufgebraucht ist oder der Spieler die 0 zur Beendigung des Glücksspiels eingegeben hat. Nach jeder Runde sollen das Würfelergebnis und das Guthaben des Spielers ausgegeben werden.

Hinweis: Achten Sie darauf, Eingaben auf ihre Gültigkeit hin zu überprüfen!

Hilfestellung: Implementieren Sie die Aufgabenstellung in kleinen Schritten so weit Sie kommen:

- Lassen Sie den Spieler zuerst nur einmal und ohne Einsatz spielen – Vergleichen Sie den Tipp mit dem Würfelergebnis!
- Lassen Sie den Spieler einen beliebigen Betrag auf eine Zahl setzen – Geben Sie den Gewinn/Verlust aus!
- Legen Sie einen Kontostand für den Spieler an und verrechnen Sie den Gewinn/Verlust mit dem Kontostand. Verhindern Sie eine Überschreitung des Kontos!
- Lassen Sie wiederholt neue Spiele mit dem aktualisierten Kontostand zu!
- Für die Dialoge können Sie die Methode `readInt(String txt)` verwenden, die ein Dialogfeld mit der Nachricht `txt` erzeugt und eine Eingabe erwartet.

Lösung 2.4

Die Lösung befindet sich in der Datei `LustigeSieben.java`.

Aufgabe 2.5 (H) Grammatik Gleitkommazahlen [4 Punkte]

Ab Java SE 7 können beliebig viele Unterstriche zwischen Ziffern in numerischen Literalen auftreten. Diese Unterstriche haben keine semantische Bedeutung und dienen lediglich der Lesbarkeit. Beispiel: `int x = 100_000;` wird interpretiert wie `int x = 100000;`.

In der Vorlesung wurde eine Grammatik für Gleitkommazahlen mit führenden Nullen angegeben. Geben Sie eine Grammatik für Gleitkommazahlen *ohne* führende Nullen (auch nicht im Exponenten) und mit den ab Java SE 7 erlaubten Unterstrichen für numerische Literale an. Beachten Sie, dass die Unterstriche nur zwischen Ziffern, aber nicht am Anfang oder Ende eines numerischen Literals, vor oder nach einem Punkt oder dem Zeichen für den Exponenten erlaubt sind, siehe `http://docs.oracle.com/javase/7/docs/technotes/guides/language/underscores-literals.html`.

Lösung 2.5

Die Lösung befindet sich in `Gleitkomma_sol.txt`.

Korrekturbemerkung:

- Ansätze, die Java-Grammatik für Gleitkommazahlen nachzubilden anstatt der in der Vorlesung gegebenen Grammatik für Gleitkommazahlen führen nicht zu Punktabzug, wenn die Bedingung, dass keine führenden Nullen (auch im Exponenten) vorkommen, erfüllt ist.
- Die Grammatik aus der Vorlesung enthält keine ganze Zahlen. Entsprechende Lösungen führen nicht zu Punktabzug.

Aufgabe 2.6 (H) Reguläre Ausdrücke [5 Punkte]

Geben Sie unter Verwendung der in der Vorlesung und im Praktikum für reguläre Ausdrücke eingeführten Operatoren *, ?, | und $^+$ reguläre Ausdrücke für die folgenden Textmuster an, wobei `letter` einen beliebigen Buchstaben im Text beschreibt:

1. Der Text enthält keine zwei aufeinanderfolgenden a's.
2. Der Text ist mindestens zwei, maximal vier Zeichen lang.
3. Der Text enthält eine positive gerade Anzahl von a's.

Es gilt `letter` $::= a|\ldots|z$.

Lösung 2.6

Die Lösung befindet sich in der Datei `l2.txt`.

Korrekturbemerkung:

1. 2 Punkt
2. 1 Punkt
3. 2 Punkte

Aufgabe 2.7 (H) Kaninchenpopulation [7 Punkte]

Auf einer einsamen Insel wird ein Paar geschlechtsreifer Kaninchen ausgesetzt, um herauszufinden, wie viele Kaninchen innerhalb eines bestimmten Zeitraums geboren werden.

Hierbei wird angenommen, dass jedes geschlechtsreife Paar jeden Monat ein neues Kaninchenpaar zur Welt bringt. Jedes Kaninchenpaar ist bereits im ersten Monat seiner Lebenszeit geschlechtsreif und jedes Kaninchen hat eine Lebenszeit von genau 3 Monaten.

- Schreiben Sie ein `MiniJava`-Programm, welches die Zahl n einliest, die den n-ten Monat bezeichnet. Ihr Programm soll dann die Anzahl der Kaninchenpaare berechnen und ausgegen, die in diesem n-ten Monat geschlechtsreif sind. Als Beispiel der Kaninchenpopulation dient folgende Tabelle:

Monat	**1. Generation**	**2. Generation**	**3. Generation**
1	1		
2	1	1	
3	2	1	1
4	4	2	1
5	7	4	2

- Gehen Sie nun davon aus, dass die Kaninchenpaare, die sich im zweiten Monat ihrer Geschlechtsreife befinden, jeweils 3 neue Kaninchenpaare zu Welt bringen. In allen anderen Generationen bringt ein Kaninchenpaar weiterhin ein neues Kaninchenpaar zur Welt.
  Schreiben Sie ein `MiniJava`-Programm, das wiederum die Zahl n einliest, die den n-ten Monat bezeichnet. Ihr Programm soll die Anzahl der Kaninchenpaare berechnen und ausgeben, die im n-ten Monat die 1., 2. und 3. Generation darstellen.

Überprüfen Sie die Eingabe auf Gültigkeit.

Lösung 2.7

Die Lösung befindet sich in den Dateien `Kaninchen.java` und `Kaninchen2.java`.

Korrekturbemerkung:

erster Teil:

- Eingabe auf Gültigkeit überprüfen: 1 Punkt
- Berechnung 2. und 3. Generation: jeweils 1 Punkt
- Berechnung 1. Generation: 2 Punkte

zweiter Teil:

- 2 Punkte

Aufgabe 2.8 (H) ggT-Tabelle [4 Punkte]

Implementieren Sie ein `MiniJava`-Programm namens `GGTTable.java`, das eine Tabelle der ggTs für alle Kombinationen von Zahlen zwischen 1 und x ausgibt (für positive x). Dabei soll x vom Benutzer eingegeben werden. Für $x = 10$ ist die Ausgabe folgende:

	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	1	2	1	2	1	2	1	2	1	2
3	1	1	3	1	1	3	1	1	3	1
4	1	2	1	4	1	2	1	4	1	2
5	1	1	1	1	5	1	1	1	1	5
6	1	2	3	2	1	6	1	2	3	2
7	1	1	1	1	1	1	7	1	1	1
8	1	2	1	4	1	2	1	8	1	2
9	1	1	3	1	1	3	1	1	9	1
10	1	2	1	2	5	2	1	2	1	10

Hinweis: In Strings können Sie \t verwenden, um einen Tabulator zu erzeugen und \n verwenden, um einen Zeilenumbruch zu erzeugen. Verwenden Sie für diese Aufgabe zur Ausgabe die Methode `System.out.println(String out)` anstatt der Methode `write(String out)`.

Z.B.

```
String tmp = "hallo, \thallo \ndu";

System.out.println(tmp);
```

Lösung 2.8

Die Lösung befindet sich in der Datei `GGTTable.java`.

Korrekturbemerkung:

- reine Berechnung des GGT gibt keine Punkte, da bekannt aus Vorlesung, falls fehlerhaft 1 Punkt Abzug
- restliche Tabelle: 4 Punkte
- bei fehlendem Tabellenrand (1 ... 10 horizontal und vertikal) 1 Punkt Abzug