technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 1
Dr. Lars Hildebrand
Iman Kamehkhosh, Marcel Preuß, Henning Timm

Praktikum zu

Einführung in die Informatik für LogWings und WiMas

Wintersemester 2013/14

Übungsblatt 2

Bearbeitungszeit:
11.–15.11.2013

Aufgabe 2.1 – Zahlensysteme

In dieser Aufgabe sollen Sie Binär- und Dezimaldarstellungen von Zahlen ineinander umrechnen, wie es auch schon in der Vorlesung demonstriert wurde. Fügen Sie in folgenden Tabellen die fehlenden Werte ein.

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Dezimalzahl
0	0	1	0	1	5
0	1	1	0	0	
0	0	0	1	0	
1	0	1	1	1	
1	1	0	0	1	

Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Dezimalzahl
1	0	0	0	0	16
					31
					26
					13
					7
					42

Aufgabe 2.2 – Primitive Datentypen und Strings

In Kapitel 1 (Folien 10–33) der Vorlesung haben Sie unterschiedliche Datendarstellungen kennengelernt. Schauen Sie sich diese Folien noch einmal an und betrachten Sie zusätzlich den folgenden Text auf der Orcale-Webseite (Link auch auf der EINI-Seite): `http://docs.oracle.com/javase/specs/jls/se7/jls7.pdf`

- Lesen Sie den Abschnitt 4.2 über primitive Datentypen. Tragen Sie dann alle primitiven Datentypen der Programmiersprache Java in der folgenden Tabelle ein und beschreiben Sie kurz in eigenen Worten, worum es sich bei dem jeweiligen Datentyp handelt.

Datentyp	Beschreibung

- Lesen Sie dann noch die Abschnitte 4.3.3 und 3.10.5 über Strings und notieren Sie, was Ihnen wichtig erscheint:

In den folgenden Aufgaben werden Sie ein einfaches Programm schreiben, das eine Temperatur von Grad Celsius nach Grad Fahrenheit umrechnen kann.

Aufgabe 2.3 – Aufbau eines Java-Programms

Legen Sie in DrJava eine neue Datei an, schreiben Sie den folgenden Quelltext ab und speichern Sie die Datei unter dem Namen „Temperatur.java“. Schreiben Sie die Zeilennummern nicht mit ab.

```
1   class Temperatur {
2    public static void main(String[] args) {
3    }
4  }
```

Ein Quelltext enthält die Beschreibung für ein Programm. Erst nach dem Compilieren entsteht das eigentliche Programm, das vom Computer ausgeführt werden kann. Compilieren Sie also nun den Quelltext und starten Sie das Programm, indem Sie die Knöpfe „Compile“ und „Run“ in DrJava benutzen[1].

Nachdem ein Programm gestartet wurde, wechselt DrJava automatisch in den „Interactions“-Tab und Sie können dort sehen, welche Ausgabe das Programm produziert hat. Die erste Zeile und alle Zeilen, die mit einem Größer-als-Zeichen beginnen, gehören jedoch nicht zur Ausgabe. Die Zeile mit „`> run Temperatur`“ zeigt an, dass DrJava das Programm „Temperatur.class“ von der Festplatte gelesen und gestartet hat (die Dateiendung „.class“ wird automatisch ergänzt und darf hier nicht angegeben werden). Das einzelne Größer-als-Zeichen in der nächsten Zeile deutet an, dass die Programmausführung beendet wurde und jetzt erneut ein Programm gestartet werden kann. Die Ausgaben Ihres Programms werden normalerweise dazwischen angezeigt, aber in dieser Teilaufgabe haben Sie ein Programm geschrieben, das keine Ausgabe produziert.

Dies ist das Grundgerüst eines jeden Java-Programms. Es ist ein „leeres“ Programm, das noch keine Funktionalität enthält.

[1] Oder drücken Sie zum Compilieren die Taste F5 und zum Starten F2.

Aufgabe 2.4 – Ausgabebefehle

Fügen Sie zwischen Zeilen 2 und 3 diese Zeilen ein:

```
System.out.print("Temperaturumrechnung");
System.out.println();
System.out.print("Die Temperatur in Grad Celsius ist: ");
```

Ab jetzt werden wir Sie nicht weiter darauf hinweisen, dass Sie den Quelltext speichern und compilieren müssen, bevor Sie das Programm ausführen können.

Führen Sie das Programm aus und achten Sie auf die Ausgabe, die Sie unter den „Interactions“ sehen können.

Aufgabe 2.5 – Kommentare

Mit zwei Schrägstrichen (//) können Sie Kommentare kennzeichnen: Der Text nach // (bis zum Zeilenende) wird beim Compilieren ignoriert. Ändern Sie die mittlere der eben eingefügten Zeilen, sodass sie wie folgt aussieht:

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

Führen Sie das Programm aus. Was ist der Unterschied? Was bewirkt die Zeile offensichtlich?

Machen Sie die Änderung rückgängig.

Aufgabe 2.6 – Variablen

Fügen Sie folgende Zeilen nach der letzten System.out.print-Zeile ein.

```
int celsius;  // Variable, die die Temperatur in Grad Celsius speichert
celsius = 20;
System.out.print(celsius);
System.out.println();
```

Führen Sie das Programm aus. Was ist hier passiert?

Aufgabe 2.7 – Umrechnung: Per Hand

Zur Umrechnung von Grad Celsius nach Grad Fahrenheit gilt diese Formel:

$$(\text{Temperatur in Grad Fahrenheit}) = (\text{Temperatur in Grad Celsius}) \cdot \frac{9}{5} + 32$$

Füllen Sie mit ihrer Hilfe die Spalte „erwartetes Ergebnis“ der folgenden Tabelle aus (die Berechnungen können Sie z.B. im Kopf oder mit einem Taschenrechner durchführen). Die Spalte „tatsächliches Ergebnis“ wird in den nächsten beiden Aufgaben ausgefüllt.

Grad Celsius	Grad Fahrenheit	
	erwartetes Ergebnis	tatsächliches Ergebnis
20		
25		
28		
42		
100		

Aufgabe 2.8 – Umrechnung: Vom Programm

Ergänzen Sie nun in ihrem Quelltext diese Zeilen (direkt nach den zuletzt eingefügten Zeilen).

```
int fahrenheit;  // Temperatur in Grad Fahrenheit
fahrenheit = celsius * 9 / 5 + 32;
System.out.print("Die Temperatur in Grad Fahrenheit ist: ");
System.out.print(fahrenheit);
System.out.println();
```

Führen Sie das Programm aus. Ist das Ergebnis wie erwartet?

Aufgabe 2.9 – Ein Bug!

Vervollständigen Sie nun die obige Tabelle, indem Sie ihr Programm für jeden Wert neu anpassen und ausführen. Was fällt Ihnen auf? Wie können Sie dies beheben?

Korrigieren Sie den Fehler.

Aufgabe 2.10 – Eine andere Fehlersorte

Ersetzen Sie nun in der Formel für die Umrechnung die `5` durch eine `0`. Compilieren Sie das Programm und führen Sie es aus. Wie unterscheidet sich der Fehler von den in Aufgabe 1.3 erzeugten Fehlern (sie müssen im „Interactions"-Tab schauen, um diesen Fehler zu sehen)?

Beheben Sie den Fehler wieder.

Aufgabe 2.11 – Umrechnung in Kelvin

Ergänzen Sie das Programm so, dass es die Temperatur auch in Kelvin anzeigt. Die Formel zur Umrechnung lautet:

$$(\text{Temperatur in Kelvin}) = (\text{Temperatur in Grad Celsius}) + 273,15$$

Nachkommastellen müssen berücksichtigt werden. Beachten Sie, dass Java in reellen Zahlen kein Dezimalkomma, sondern einen Dezimalpunkt erwartet. Testen Sie das Programm für den Wert 5 Grad Celsius.

Aufgabe 2.12 – Vereinfachungen

Wenn Sie mehrere System.out.print-Befehle verwenden, z. B. auf diese Art:

```
System.out.print("Eine tolle Zahl ist: ");
System.out.print(x);
System.out.println();
```

So kann der Quelltext folgendermaßen vereinfacht werden:

```
System.out.println("Eine tolle Zahl ist: " + x);
```

Vereinfachen Sie nach diesem Schema Ihren Quelltext.

Ergänzende Aufgaben

Aufgabe 2.13 – Notwendigkeit verschiedener Datentypen

Alle Daten im Computer repräsentieren sich durch Reihen von Nullen und Einsen — s.g. Bitfolgen. Das betrifft Zahlen, Bilder, Musik sowie Programme und das Betriebssystem selbst. Allein die Datentypen können eine Aussage darüber treffen, wie eine bestimmte Bitfolge interpretiert werden soll. So kann ein und dieselbe Bitfolge je nach Datentyp etwas völlig anderes darstellen.

Versuchen Sie anhand folgender Tabelle herauszufinden, um welche Bitfolge es sich dabei handeln könnte:

Datentyp	Entsprechender Wert
char	F
byte	48 *(letzte 8 Bit)*
short	-22736
int	42800
long	42800
float	$5,9976 \cdot 10^{-41}$
double	$2,1146 \cdot 10^{-319}$