Institut für Informatik
Praktische Informatik und Bioinformatik
Prof. Dr. Ralf Zimmer

Wintersemester 2019/20

Übungen zum Propädeutikum Programmierung in der Bioinformatik

Blatt 1

Termin: Dienstag, 22. Oktober 2019

Lösung 2

```
        int     tag = 22;
        int     monat = 10;
        int     jahr = 2019;
        String  wochentag = "Dienstag";
        boolean regnet_es = false;
        char    ersterBuchstabe = 'A';
```

Lösung 3

```
String name = "Max Mustermann";
int alter = 19;
double groesse = 1.85;
System.out.println("Name: " + name + "\n" +
                   "Alter: " + alter + "\n" +
                   "Größe: " + groesse);
```

Output:

```
Name: Max Mustermann
Alter: 19
Größe: 1.85
```

Lösung 4

```
public class Main {
    public static void main(String[] args){
        int alter = 18;
        String name = "Max Mustermann";

        if (alter < 18) {
            System.out.println("Sorry " + name + ", du bist leider zu jung.");
        } else {
            System.out.println("Ready to Go, " + name + "!");
        }
    }
}
```

Output:

```
Ready to Go, Max Mustermann!
```

Lösung 5

1.
```
Hello there
```

2.
```
c
```

3.
```
a ungleich d
```

Lösung 6

Zunächst testen wir den Output den Java uns gibt auf zwei verschiedene Arten:

```
1 // Versuch 1: Wir probieren es ohne vorher die Typen der Variablen festzulegen
2 System.out.println((5/2));
3 // Versuch 2: Wir antizipieren, dass das Ergebnis ein Double sein sollte
4 int num1 = 5;
5 int num2 = 2;
6 double result = num1 / num2;
7 System.out.println(result);
```

Wider Erwarten ist das Ergebnis nicht `2.5` sondern `2` für den ersten Versuch, bzw. `2.0` für den zweiten Versuch. Warum ist das so? Beim dividieren von `int`s, wird eine *Integer Division* durchgeführt. Das bedeutet, dass Java das Ergebnis truncated. Einfach gesagt: Java schneidet alles ab was hinter dem Komma kommen würde.

Warum wird dann der zweite Versuch trotzdem als `2.0` ausgegeben? Das kommt daher, dass in Zeile 6 des obigen Codeblocks zunächst der Teil *hinter* dem `=` ausgewertet wird, also `5 / 2`. Wie wir aus Versuch 1 wissen, wird das zu einer `2` ausgewertet. Da wir Java nun aber im Teil *vor* dem `=` sagen, dass dieses Ergebnis als `double` abgelegt werden soll, wird diese `int`-Zahl automatisch von Java zu einem `double` konvertiert. Somit wird die ganze Zahl `2` zu einer `2.0`.

Das eigentliche Problem lässt sich z.B. lösen indem man vorher eine der Zahlen zu einem `double` konvertiert. Sobald ein Teil der Operanden ein `double` ist, verwendet Java keine Integer Division mehr. Folgende Beispiele ergeben alle `2.5`:

```
1  int num1 = 5;
2  int num2 = 2;
3  // Option 1: Typecasting
4  double result1 = (double)num1 / num2;
5  System.out.println(result1);
6  System.out.println((5 / (double)2));
7  // Option 2: Multiplikation mit einem double
8  double result2 = (num1 * 1.0) / num2;
9  System.out.println(result2);
10 System.out.println((5 / (2 * 1.0)));
```

Typecasting, also eine Typumwandlung, von `int` zu `double` ist möglich indem man einfach ein `(double)` vor die Zahl schreibt, die als `double` interpretiert werden soll. Die Multiplikation mit `1.0` hat den selben Effekt. Man muss allerdings beachten, dass Typecasting nicht von jedem Typ zu jedem anderen Typ funktioniert.

Lösung 7

Hierbei ist wichtig sich zu überlegen wie man die Aufgabe in eine sinnvolle Programmstruktur übersetzt. Wir wissen, dass das Programm vier verschiedene Grundrechenarten unterscheiden können muss. Welche Operation verwendet wird, entscheidet der Operator. Dementsprechend ist es sinnvoll das Programm auch über eine Entscheidung zu modellieren, was mit if-else-Konstrukten möglich ist, in denen überprüft wird ob unsere `operator`-Variable gleich einem der vier Fälle ist:

```java
public class Taschenrechner {
    public static void main(String[] args) {
        int num1 = 5;
        int num2 = 5;
        char operator = '*';

        if (operator == '+') {
            System.out.println(num1 + " + " + num2 + " = " + (num1 + num2));
        } else if (operator == '-') {
            System.out.println(num1 + " - " + num2 + " = " + (num1 - num2));
        } else if (operator == '*') {
            System.out.println(num1 + " * " + num2 + " = " + (num1 * num2));
        } else if (operator == '/') {
            System.out.println(num1 + " / " + num2 + " = " + (num1 / (double) num2));
        } else {
            System.out.println("Kein gültiger Operator");
        }
    }
}
```

Besonders zu beachten sind hier zwei Dinge. Man kann solche if-else-Konstrukte hintereinanderschalten um unkompliziert mehrere Fälle zu überprüfen. Außerdem ist die Verwendung des letzten `else` besonders interessant, da man hier alle Fälle die durch unseren “Filter” gefallen sind “abfangen” kann. Somit lassen sich beim Taschenrechner leicht unerwartete oder fehlerhafte Eingaben abfangen und behandeln.