Allgemeine Übungshinweise

A. Alle Aufgaben sind mit * gekennzeichnet und zeigen die (subjektive) Einschätzung der Schwierigkeit und des Aufwands der Aufgabe:
   - ( ) besonders leichte Aufgabe; Bearbeitungsdauer < 10 min
   - (*) einfache Aufgabe, die auch von Anfängern leicht bewältigt werden kann
   - (**) Aufgabe mit einem durchschnittlichen Arbeitsaufwand
   - (***) kompliziertere Aufgabe, die einen deutlich höheren Aufwand erfordert

   Außerdem sind die Themengebiete der Aufgabe in kursiver Schrift markiert.

B. Ab Übungsblatt 2 gibt es Zusatzaufgaben, die für diejenigen gedacht sind, die die Aufgaben sehr schnell bearbeiten, Vorkenntnisse haben und/oder weitere Aufgaben üben möchten. Diese Zusatzaufgaben vertiefen auch z.T. einzelnes Wissen und erweitern ggf. das Programmieren 1 Themengebiet.

C. Alle Aufgaben, die nicht als Zusatzaufgaben gekennzeichnet sind, sind zum Erlernen der damit verbundenen Grundlagen gedacht und daher als Muss-Aufgabe zu betrachten. Die Bearbeitung dieser Muss-Aufgaben wird in der Klausur vorausgesetzt.

Übungen Kapitel 1

(Nur) für dieses Übungsblatt 1 gilt:
- Verwenden Sie hierzu zunächst NICHT die Eclipse Umgebung, sondern
- Erstellen Sie die Programme mit `notepad` als Editor
- Lassen Sie das Programm in der Kommandozeile über `java` laufen. Beachten Sie Installationshinweise im moodle Kurs (INSTALLATION_README_FIRST Datei).

1.1 ( ) The first java program

Tippen Sie das in der Vorlesung vorgestellte „Welcome“-Programme (Skript S. 1-47) ab (oder copy-paste), kompilieren es auf der Console mit `javac` und führen es anschließend mit `java` aus. Machen Sie sich alle Schritte, die Sie getan haben, klar!

1.2 (*) Syntaxprüfung *(Compiler, Parser)*

1. Leiten Sie folgendes Programm aus der in der Vorlesung (Skript S. 1-17ff) vorgestellten Grammatik in EBNF Notation her. Notieren Sie dabei jeden einzelnen Schritt sowie die jeweils angewandte Regel:

```
PROGRAM HELLO
  BEGIN
    J1    := 56;
    S     :="This is a program";
    A45   := 1000;
  END;
```

2. Erklären Sie an diesem Beispiel die Begriffe Schlüsselwort und Bezeichner.

1.3 ( ) Begriffe *(Compiler, Interpreter)*

1. Erklären Sie den Unterschied zwischen einem Interpreter und Compiler.
2. Wahr oder falsch: "Java ist eine interpretative Sprache".
   Begründen Sie Ihre Antwort

1.4 (*) Ausgabe auf der Console *(Ausgabe mit `System.out`)*

Was druckt der folgende Code aus:

a.
```
System.out.print("*\n**\n***\n****\n*****");
```

b.
```
System.out.println("*");
System.out.println("***");
System.out.println("*****");
System.out.println("****");
System.out.println("**");
```

c.
```
System.out.print("*");
System.out.print("***");
System.out.print("*****");
System.out.print("****");
System.out.println("**");
```

d.
```
System.out.print("*");
System.out.println("***");
System.out.println("*****");
System.out.print("****");
System.out.println("**");
```

Machen Sie sich bei dieser Aufgabe mit der Dokumentation der Java API für das Objekt `out` der Klasse `System` vertraut. Informieren Sie sich über einzelnen Methoden, insbesondere über `println` und `print`.

Lösungshinweis:
API ist ein Akronym für Application Program(ming) Interface. Die API beschreibt die verfügbaren Sprachelemente, so dass ein Programmierer sie benutzen kann. Nutzen Sie für die online Dokumentation das Web z.B. http://docs.oracle.com/javase/8/docs/api/

1.5 (*) Java Syntaxfehler *(Block, Variable, Zuweisung)*

Korrigieren Sie die Fehler folgendes Programms:

```
1     public class Summe
2     {
3           public static main ( String [ ] args)
4           {
5                 Int summe
6                 int summand1 = 12.5 ,
7                 Int summand2 = 24 ;
8
9                 { summe = summand1 + summand2 ; }
10                ;
11                System . out . print (Summand1 ) ;
12                System . out . print ( "+" ) ;
13                System . out . print (Summand2 ) ;
14                System . out . print( '=' ) ;
15                System . out . println(summe ) ;
16
17
18              System . out . print (Noch e i n e andere Summenausgabe:) ;
19                System . out . println (summe ) ;
20                 }
```

Lösungshinweis: Halten Sie sich an die bislang vorgestellten wenigen Java Syntaxwörter.

1.6 (**) Eingabe bei der Ausführung *(Variable, Arithmetik, `main` Argument `args`)*

Beim Ausführen eines Java-Programms ist es möglich, Parameter mit zu übergeben. Betrachten wir folgendes Programm (aus der Vorlesung), das die Summe `1+2+3+ … +n` berechnet:

```
class Sum {
     public static void main(String[] args)  {
          int n = 5;
          int result = 0;
          int counter = 1;
          while(counter <= n)      {
               result = result + counter;
               counter = counter + 1;
          }
          System.out.println(result);
     }
  }
```

Dieses Programm hat einen wesentlichen Nachteil: es berechnet die Summe für den Wert `n = 5`. Was machen wir aber, wenn wir die Summe für den Wert z.B. `n=25` berechnen müssen? Wir müssen die (rote) Zeile auf **`int n = 25`** ändern! Dies ist sehr änderungsunfreundlich, denn für jeden anderen Wert von n müssen wir den Source Code ändern.
In Java gibt es nun die Möglichkeit, beim Aufruf eines Programms dem Programm Werte mitzugeben:

```
E:\ java Sum 25
>325

E:\ java Sum 15
>120
```

Dazu muss unser Programm wie folgt geändert werden:

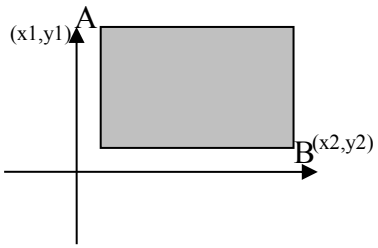
```
class Sum {
    public static void main(String[] args)  {
        int n = Integer.parseInt(args[0]);

        int result = 0;
        int counter = 1;
        while(counter <= n)      {
            result = result + counter;
            counter = counter + 1;
        }
        System.out.println(result);
    }
}
```

Jetzt bleibt der Source Code gleich und unser Programm kann beliebige (ganzzahlige) Werte für n verarbeiten!

Falls beim Aufruf mehr als ein Wert mitgegeben werden soll, können Sie mit `args[0]` auf den ersten, mit `args[1]` auf den zweiten, mit `args[2]` auf den dritten Wert usw. zugreifen. `Integer.parseInt` wird später erklärt. Schreiben Sie es einfach immer so ab!

a. Testen Sie das `Sum` Programm mit der Eingabe beim Aufruf für verschiedene n's.
b. Implementieren Sie eine Klasse `RectangleArea`, das die Fläche eines Rechtecks berechnet. Ein Rechteck ist dabei wie folgt definiert:



Für das Rechteck A(1,5), B(5,1) berechnet das Programm die Fläche 16:

```
E:\ java RectangleArea 1 5 5 1
>16
```

Lösungshinweis: Sie dürfen der Einfachheithalber bei der Aufgabe annehmen, dass das Rechteck nur im positiven Quadranten liegt und die Koordinaten immer ganzzahlig sind.

1.7 (*) Multiplikation *(Variablen, Arithmetik, `while` Schleife)*

Berechnen Sie die Multiplikation 3*4*5* ... * n = ?, wobei n >=3 ist. Für n = 3 ist das Resultat 3. Für alle Werte n < 3 ist das Resultat 1. Gehen Sie hierzu in den vorgestellten Schritten vor und erstellen Sie ein Java Programm. Dabei orientieren Sie sich an dem Program `Sum`. Für die Multiplikation verwenden Sie das Java Symbol `*`. Den Wert n geben Sie wieder bei der Ausführung mit:

```
E:\ java Mult 6
>360

E:\ java Mult 3
>3

E:\ java Mult 0
>1
```

1.8 (*) Code Conventions *(Syntax Kommentar, Escape Sequenzen)*

Gegeben ist folgendes Java Programm in der Datei Konventionen.java:

```
// Example: konventionen.java
   Ausgabe der URL für die Java Code Conventions
  */ /* Klasse konventionen

  class konventionen

  public void Main(string s[]) {
  System.out.print("You will); System.out.println(" find ")
  System.out.println(Java Code Conventions  ); }
  System.out.println("under  \\t"); /** position to the next
tab **/
  System.out.println(""http://java.sun.com/docs/codeconv/ "");
  }
```

Finden Sie in folgendem Code Stück die Fehler und erstellen ein lauffähiges Programm. Dabei schreiben Sie ihr korrigiertes Programm exakt nach den bislang vorgestellten Programmierkonventionen mit `notepad` auf.

1.9 (*) Power *(Variablen, Arithmetik, `while` Schleife)*

Erstellen Sie ein Programm `Power`, das für die positiven Zahlen 1 bis n jeweils das Quadrat und die 3-er Potenz berechnet. Für die Multiplikation verwenden Sie wieder `*` und nur die bislang bekannten Java Konstrukte. Beispielsweise sieht das Ergebnis für n = 3 und n = 5 so aus:

```
E:\ java Power 3
     >Zahl    Quadrat Kubisch
     >======================
     >1       1       1
     >2       4       8
     >3       9       27

   E:\ java Power 5
     >Zahl    Quadrat Kubisch
     >======================
     >1       1       1
     >2       4       8
     >3       9       27
     >4       16      64
     >5       25      125
```

1.10 (*) Eingabe durch Benutzer *(Scanner Klasse)*

Ändern Sie Aufgabe 1.7 so um, dass der Wert n durch die Eingabe des Benutzers aus der Scanner Klasse erfolgen. Hierzu verwenden Sie die einzelne Codefragmente aus `Codebeispiele\ScannerExample`. Schauen Sie sich das Codebeispiel `ScannerExample` an und kopieren das Codeschnipsel in Ihr Programm.
Eine mögliche Ausgabe sieht so aus:

```
Bitte n eingeben
4
12
```

Lassen Sie ihr Programm mehrmals laufen und geben jeweils unterschiedliche Werte für n ein: z.B. 3 , 4, 15 und 30. Wie erklären Sie sich das Resultat bei 30?