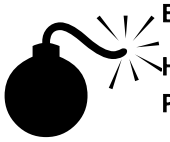

Erstellen Sie für JEDES Programm, bevor Sie dieses am Computer erstellen, das Stuktogramm.

Halten Sie beim Programmieren rigoros ALLE aufgestellten Programmierregeln ein. Ihre Programme werden in Hinsicht auf die Einhaltung dieser Regeln kontrolliert.

1. Sie sollen das nachfolgende Programm übersichtlich gestalten indem Sie die aufgestellten *Programmierregeln* nebst *Einrückungen* beachten. Die im Programm vorhandenen *Kommentare* sollen Sie wegstreichen.

```
/** * @author Sepp */ public class EineLangeWurst { /** * @param args */
public static void main(String[] args) { int i = 0; int n = 60;
System.out.println("Ergebnis"); while (i <= n) { if (i % 7 == 5) {
System.out.println(i); } i = i + 1; } } }
```

2. Das nachfolgende Programm sollen Sie ebenfalls nach den aufgestellten *Programmierregeln* gestalten. Zudem sollen Sie hier die *Groß-/Kleinschreibung* richtig anwenden (**ACHTUNG**: Starten Sie das Programm nicht!!!).

```
PUBLIC CLASS SYNTAXSEMANTIK { PUBLIC STATIC VOID MAIN(STRING[] ARGS) { INT N = 5;
INT I = 0; WHILE (I <= N) { INT J = 0; WHILE (J <= I) { SYSTEM.OUT.PRINT(J); }
SYSTEM.OUT.PRINTLN(); } } }
```

3. Das obige Programm soll die nebenstehende Ausgabe produzieren. Ändern Sie das Programm entsprechend ab (**Hinweis**: Die Methode `println` bewirkt bei der Ausgabe zum Unterschied zur Methode `print` einen Zeilensprung).

```
0
01
012
0123
01234
012345
```

4. Betrachten Sie folgendes Programm und formulieren Sie es so um, so dass es dasselbe Ergebnis liefert aber vom Gesichtspunkt der *Pragmatik* leichter verständlich und nachvollziehbarer ist.

```
public class Pragmatik
{
   public static void main(String[] args) {
      int n = 10;
      int i = -3;
      while (i + 4 < n + 1) {
         System.out.println(i + 4);
         i = i + 2;
      }
   }
}
```

5. Suchen Sie im nachfolgenden Programm die *Fehler*, und kennzeichnen Sie diese. Beschreiben Sie jeden Fehler und beschreiben Sie, wie es richtig gemacht werden müsste.

```
public Fehler
{
   public void main(string[] args) {
      int summe = 0;
      int n = 10;
      int j = n / 3;
      while (i <= n) {
         summe = summe + i
         i = i + j;
      }
   }
   System.out.println(summe);
```

6. Bringen Sie das übersichtlich gestaltete Programm der ersten Aufgabe zum Laufen. Sie sollen dabei nachvollziehen und verstehen, was das Programm macht. Beachten Sie dabei dass der %-Operator den Rest zweier Ganzzahlen ermittelt.
   So ergibt beispielsweise 12 % 7 das Ergebnis 5.

7. Betrachten Sie die Größen der in der vorigen Aufgabe abgespeicherten *Quellcode-Datei* `EineLangeWurst.java` und des *Bytecodes* in der Datei `EineLangeWurst.class`. Welche Datei ist größer?

8. Schreiben Sie dann ein Programm das den Namen `SummeSieben` hat, das alle *Vielfachen von 7* unter 1000 addiert und die Summe ausgibt. Wie können Sie nachprüfen, ob Ihr Programm richtig arbeitet?

9. Schreiben Sie weiters ein Programm mit dem Namen `Fakultaet`, welches die *Fakultät* von n – geschrieben als *n!* – anhand folgender Vorschrift ermittelt:

   $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-2) \cdot (n-1) \cdot n$ für $n \geq 1$

   Testen Sie Ihr Programm für `n = 6`. Das korrekte Ergebnis ist 720.

10. Die sogenannte *Doppelfaktorielle* von n ist definiert als

    $1 \cdot 3 \cdot 5 \cdot \ldots \cdot (n-4) \cdot (n-2) \cdot n$ für ungerade $n \geq 1$ und
    $2 \cdot 4 \cdot 6 \cdot \ldots \cdot (n-4) \cdot (n-2) \cdot n$ für gerade $n \geq 2$.

    Entwickeln Sie ein Programm mit dem Namen `DoppelFaktorielle`, das die *Doppelfaktorielle* einer Zahl n berechnet und ausgibt. So soll das Programm für `n = 10` das Ergebnis 3840 (= 2·4·6·8·10) ausgeben.

11. Ersetzen Sie im vorigen Programm die Anweisung `int n = 10;` durch

    ```
    int n = Integer.parseInt(args[0]);
    ```

    Diese neue Anweisung holt sich beim Programmstart den Wert für n von der Kommandozeile. Beim Starten des Programms können Sie beispielsweise den gewünschten Wert von n hinter dem Programmnamen angeben:

    ```
    java DoppelFaktorielle 10
    ```

12. Schreiben Sie ein Programm mit dem Namen `Teiler`, welche alle *Teiler* einer über die Kommandozeile eingegebenen ganzen Zahl z ermittelt.
    So müsste das Programm für `z = 10` die Ergebnis 1, 2, 5 und 10 liefern.

13. Schreiben Sie ein Programm mit dem Namen `PrimzahlTest`, welches eine über die Kommandozeile übergebene Zahl z testet und kontrolliert, ob diese *Primzahl* ist. Das Programm soll so geschrieben werden, dass sofort beim Erkennen dass die Zahl keine Primzahl ist, die Schleife abgebrochen und der Text *„Die Zahl ist keine Primzahl“* ausgegeben wird. Wenn erkannt wird, dass die Zahl Primzahl ist, soll der Text *„Die Zahl ist Primzahl“* ausgegeben werden.

14. Schreiben Sie weiters ein Programm mit dem Namen `EinMalEins`, welches eine *Einmaleins-Tabelle* in der angegebenen Form ausgibt. Dabei soll auch die Überschrift exakt laut Vorlage ausgegeben werden.

    Wichtig: Die Ausgaben sollen anhand zweier ineinander geschachtelter `while`-Schleifen erfolgen.

    Damit die Zahlen rechtsbündig untereinander ausgegeben werden, sollen Sie sich folgender *Methode* `printZahl` bedienen, welche Sie in Ihr Programm integrieren sollen:

```
Einmaleins-Tabelle
==================
   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
   4   8  12  16  20  24  28  32  36  40
   5  10  15  20  25  30  35  40  45  50
   6  12  18  24  30  36  42  48  54  60
   7  14  21  28  35  42  49  56  63  70
   8  16  24  32  40  48  56  64  72  80
   9  18  27  36  45  54  63  72  81  90
  10  20  30  40  50  60  70  80  90 100
```

```
/**
 * Gibt die ihr übergebene Zahl rechtsbündig auf vier Stellen aus. So wird
 * beispielsweise die Zahl 5 folgendermaßen ausgegeben: ...5 während die
 * Zahl 100 so ausgegeben wird .100
 * @param zahl die auszugebende Zahl
 */
public static void printZahl(int zahl) {
   if (zahl < 10)
      System.out.print("..." + zahl);
   else
      if (zahl < 100)
         System.out.print(".." + zahl);
      else
         System.out.print("." + zahl);
}
```

Um die Methode *aufzurufen*, gehen Sie folgendermaßen vor:

```
printZahl(12);
```

15. Schreiben Sie weiters ein Programm mit dem Namen FakultaetTabelle, welches die Fakultät der Zahlen von 1 bis 10 in *Tabellenform* wie angegeben ausgibt.

    Wichtig: Die Ausgaben sollen anhand zweier ineinander geschachtelter `while`-Schleifen erfolgen.

    Auch der Tabellenkopf soll exakt wie angegeben ausgegeben werden.

    Zur rechtsbündigen Ausgabe der Zahlen sollen Sie die vorhin eingeführte *Methode* `printZahl` erweitern, so dass die Zahl auf 8 Stellen rechtsbündig ausgegeben wird. So wird beispielsweise die Zahl 720 mit 5 vorangehenden Leerzeichen (.....720) ausgegeben.

```
       n       n!
================
       1        1
       2        2
       3        6
       4       24
       5      120
       6      720
       7     5040
       8    40320
       9   362880
      10  3628800
```

16. Schreiben Sie dann weiters ein Programm, welches für alle Zahlen zwischen 90 und 120 ausgibt, ob sie Primzahlen sind oder nicht. Das Programm soll den Namen `PrimzahlenTabelle` haben.

    In diesem Programm sollen Sie sich eine eigene Methode `istPrimzahl` *deklarieren*, *programmieren* und dann *aufrufen*, welche für die ihr übergebene Zahl ausgibt, ob sie Primzahl ist (*„ist Primzahl"*) oder nicht (*„ist nicht Primzahl"*).

    Wählen Sie folgenden *Methodenkopf*:

    ```
    public static void istPrimzahl(int z) { … }
    ```

    Wird die Methode dann beispielsweise folgendermaßen *aufgerufen*

    ```
    istPrimzahl(5);
    ```

    so soll der Text `ist Primzahl` ausgeben werden.

    Die Zahl selbst soll – wie in der Wertetabelle ersichtlich – rechtsbündig auf 4 Stellen ausgegeben werden. Verwenden Sie dazu die bereits bekannte Methode `printZahl`, und integrieren Sie diese in Ihr Programm.

```
Primzahlentabelle
=================
  90 ist nicht Primzahl
  91 ist nicht Primzahl
  92 ist nicht Primzahl
  93 ist nicht Primzahl
  94 ist nicht Primzahl
  95 ist nicht Primzahl
  96 ist nicht Primzahl
  97 ist Primzahl
  98 ist nicht Primzahl
  99 ist nicht Primzahl
 100 ist nicht Primzahl
 101 ist Primzahl
 102 ist nicht Primzahl
 103 ist Primzahl
 104 ist nicht Primzahl
 105 ist nicht Primzahl
 106 ist nicht Primzahl
 107 ist Primzahl
 108 ist nicht Primzahl
```

    Die Struktur des Programms soll in etwa folgendermaßen aussehen:

```
public class PrimzahlenTabelle
{
   public static void main(String[] args) {
      int zahl = 90;
      int n = 120;
      while (zahl <= n) {
         printZahl(zahl);
         istPrimzahl(zahl);
         zahl = zahl + 1;
      }
   }

   public static void istPrimzahl(int zahl) {
      …
   }

   public static void printZahl(int zahl) {
      …
   }
}
```