Java Arbeitsblatt 1

1 Hello World (IDE, Terminal, erstes Programm, Parameter, Strings)

1. Erstelle im Ordner *Projects* einen Unterordner *Arbeitsblatt_1*
2. Öffne Visual Studio Code und öffne (Ctrl&K, Ctrl&O) den Unterordner *Arbeitsblatt_1*
3. Erstelle eine Datei mit dem Namen *Hello.java* (Gross- und Kleinschreibung beachten!) und fülle sie mit dem folgenden Code:
```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello KSL");
    }
}
```
(Tabstopps für Einrückungen!)
Speichere (Ctrl&S)
4. Öffne eine Konsole (Terminal/ New Terminal) und tippe darin die folgenden Befehle (jeweils mit Enter bestätigen):
`javac Hello.java` (Compiliert die Datei zu *Hello.class*)
`java Hello` (führt die Datei *Hello.class* aus)
Kontrolliere mit `ls` oder `dir`, welche Dateien im Verzeichnis sind.
weiterer Grundbefehl: `cd ..` bzw. `cd Grundlagen` (change directory)
5. Ändere die `main`-Methode zu:
```
public static void main(String[] args) {
String name = args[0];
System.out.println("Hello " + name);
}
```
Compiliere die Datei und führe sie mit `java Hello Xyz` (ersetze Xyz durch deinen Namen) aus.

2 Calc (Integers Integer-Operationen, Integer-Parameter)

1. Erstelle eine Datei *Calc.java* mit dem Inhalt
```
public class Calc {
    public static void main(String[] args) {
        int a = 34;
        int b = 7;
        System.out.println("result: " + a + b);
    }
}
```
Compiliere sie und führe sie aus (Abkürzung: „Run“-Knopf über main-Methode). Offenbar wurden `a` und `b` in Strings umgewandelt und an den String `"result: "` angehängt. Dies nennt man automatische Typkonversion.

2. Versuche, als Resultat die Summe von `a` und `b` angezeigt zu bekommen. (Lösung in der Endnote)[i]

3. Ersetze `+` der Reihe nach durch `-`, `*`, `/` und `%` und beobachte das Verhalten. Falls nötig, experimentiere mit anderen Werten für `a` und `b`.[ii]

4. Ersetze die beiden Zahlen im Code mit `Integer.parseInt(args[0])` und `Integer.parseInt(args[1])`. Compiliere danach das Programm (wieder im Terminal) und führe es aus als `java Calc 34 7` (oder mit beliebigen anderen Zahlen).

3 Powers (Methoden, boolean, Verzweigungen)

1. In der Datei *Powers.java* ruft die `main`-Methode die Methode `square` auf. Dabei übergibt sie den Integer `base` als Argument und bekommt als `return`-Wert wieder einen Integerwert zurück. Deshalb muss `int` (=der Typ des Rückgabewertes) vor der Definition der Methode `square` stehen. Wenn eine Methode nichts zurückgibt (wie die `main`-Methode), steht `void`.
   - `static` muss im Moment vor jeder Methode stehen, wir werden später sehen, wieso.
   - `//` und der `/** * **/`-Block sind Kommentare, die erklären, was der Code tut. Sie haben keinen Einfluss darauf, was der Computer tut.

2. Ergänze die Datei mit einer zweiten Methode `cube(int base)`, die die dritte Potenz des Arguments `base` zurückgibt. Teste sie mit einer Ausgabe in der Konsole.

3. Probiere aus, was `cube(10000)` gibt. Offenbar gibt Ihre Methode für Argumente grösser als 8470 Quatsch aus. **Fortgeschrittenen-Aufgabe:** Erkläre möglichst genau, warum und wie das falsche Resultat zustande kommt.

4. Wir wollen eine Warnung auf der Konsole ausgeben, falls `base` grösser als 8470 ist. Ergänzen Sie vor dem `return`-Befehl in der Methode `cube` den Code
```
if (base > 8470) {
    System.out.println("Warning: The cube of " + base + " is outside of the range of int.");
}
```
Probieren Sie aus.

5. Ersetze die Klammer `}` des `if`-Befehls durch
```
} else if (base == 0 || base == 1) {
 System.out.println("useless operation, but ok");
} else {
 System.out.println("Nothing to worry about :-)");
}
```
und probiere aus. Du hast eine sogenannte Verzweigung programmiert.

   - Die Bedingung `(...)` für das Ausführen der geschweiften Klammer ist vom Typ `boolean` (mögliche Werte: `true` oder `false`). Boolesche Werte können in Variablen gespeichert werden mittels `boolean a = true`; oder `boolean bedingung = (a >= 0)`; Sie können dann benutzt werden in Ausdrücken wie `if a {...}`.
   - Etwas boolesche Logik:

`&&`	und (AND)	`==`	ist gleich	`<`	ist kleiner als
`||`	oder (OR)	`!=`	ist ungleich	`>=`	ist grösser oder gleich
`!a`	nicht `a` (NOT)	`>`	ist grösser als	`<=`	ist kleiner oder gleich

     Ausserdem kann mit Klammern gearbeitet werden:
     `(a > b && !(a <= 0)) || a == 0`
   - Es können beliebig viele `else if (...) {...}`-Blöcke aneinandergereiht werden. Der Block kann auch weggelassen werden.

4 Fortsetzung von Powers (Schleifen)

1. Ergänze deine Klasse `Powers` mit der folgenden Methode:
```
static int power(int base, int n) {
    int power = 1;
    int i = 0; //counter
    while (i < n) { //repeats what is in {} as long as () is true
        power = power * base;
        i++; //short for "i = i + 1";
    }
    return power;
}
```
   Schreibe wieder Testcode, um `power(a, b)` zu testen.

2. Statt einer `while`-Schleife gibt es auch noch die `for`-Schleife. Der folgende Code tut genau das Gleiche:
```
static int power(int base, int n) {
    int power = 1;
    for (int i = 0; i < n; i++) {
        power = power * base;
    }
    return power;
}
```

i Lösung 1: Klammern um `a + b`.
Lösung 2: `int c = a + b;`
`System.out.println("result: " + c);`

ii `34 / 7 = 4` ganzzahlige Division mit Rest
`34 % 7 = 6` Rest (=Modulus) der ganzzahligen Division `34 / 7`.