Minimales Java Konsolen Programm und wie man es von der Kommandozeile startet

Ein Java Programm ist innerhalb einer IDE (Programmierumgebung) ist schnell geschrieben und ausgeführt. Aber oft sieht man nicht was dabei eigentlich tatsächlich passiert. Spätestens wenn man dann sein Programm mal gerne selbst auf der Kommandozeile (CLI) ausführen möchte fangen die Probleme an.

Dieser Artikel soll den ganzen Prozess mal auf unterster Ebene, angefangen vom Source Code bis hin zum ausführbaren Java Programm zeigen.

Hier mein Beispielprogramm:

ListArgs.java

```
public class ListArgs {

    public static void main(String[] args)    {

        if(args.length=0){
            println("No Args found.");
        } else {
            println("Following Args been found:");
            int argCounter=0;
            for (String string : args) {
                println(argCounter + " : " + string);
                argCounter++;
            }
        }
    }

    private static void println(String string) {
        System.out.println(string);
    }

    private static void print(String string) {
        System.out.print(string);
    }

}
```

Was macht das Programm?

Nicht viel. Es überprüft ob dem Programm “Argumente” bzw “Parameter” mit übergeben wurden und listet diese dann einfach nur auf. Wurden keine Argumente übergeben meldet es uns auch das.

Übrigens hat die **zweite Zeile** eine besondere Bedeutung:

```
public static void main(String[] args) {
```

Das ist der “Einstiegspunkt” oder “Startpunkt” unseres Programms. An dieser Stelle ist es wichtig dass es eine “Methode” (unter C/C++ nennt man es eher Funktion) mit den Namen “main” gibt welche zwingend ein String-Array aufnehmen kann. Nur dann wird der Compiler später auch von sich aus erkennen können welches Programm bzw welche Methode er zu starten hat.

Wie bringe ich das Programm zur Ausführung?

Ganz so schnell gehts nun doch nicht. Noch liegt uns das Programm nur als “Source Code” vor. Source Code ist nicht ausführbar. Bevor wir es ausführen können müssen wir es erst noch kompilieren (to compile).

Zur Ausführung brauchen wir also:

1. einen Java Compiler - zur Wandlung von Sourcecode in sog. Byte-Code (manchmal auch Objekt-Code oder Binär-Code genannt).
2. eine Java Virtual Machine (manchmal auch Java VM, JVM oder JRE genannt). Diese kann dann den Byte-Code ausführen.

Sollten wir mal KEINEN Source Code vorliegen haben sondern statt dessen nur die *.class Datei (Byte Code), so brauchen wir keinen Compiler mehr sondern nur noch eine JRE um das Programm auszuführen. In unserem Beispiel müssen wir diese allerdings durch kompilieren erst noch selbst erzeugen.

Der Java Compiler ist Bestandteil des Oracle Java SDK oder auch JDK genannt. Dieses Softwarepaket gibts für die unterschiedlichsten Betriebssysteme. Es enthält neben dem Compiler auch gleich noch eine JRE, also alle Komponenten um unser Java Programm später ausführen zu können, sowie jede Menge Bibliotheken und Dokumentation. Dieses JDK gibts u.a. bei Oracle auf www.oracle.com unter der Rubrik “Java”.

Ein JRE gibts dort ebenso, wird aber wie bereits erwähnt nur dann benötigt wenn auf dem Zielsystem kein SDK installiert wurde und wir bereits Byte Code, also *.class Dateien vorliegen haben.

Wie kompiliere ich Java Source Code?

Das ist im einfachsten Fall (unserem) nicht sehr schwer.

1. öffnen Sie eine Kommandozeile (cmd.exe)

2. wechselt der Einfachheit halber in das Verzeichnis in welche Sie *.java Source Code Datei gespeichert haben
3. rufen Sie wie gezeigt den Java Compiler mit den gezeigten Parametern auf und hängen Sie dahinter den Dateinamen der Source Code Datei. Achten Sie dabei auf die Pfadangaben zum Compiler. Diese können bei Ihnen abweichend sein. Einfach entsprechend anpassen.

```
"c:\Program Files\Java\jdk1.6.0_24\bin\javac.exe" -verbose -deprecation
ListArgs.java
```

4. Wenn alles soweit richtig ist zeigt der Compiler in etwa folgendes an:

```
[parsing started ListArgs.java]
[parsing completed 22ms]
[search path for source files: .]
[search path for class files: c:\Program
Files\Java\jdk1.6.0_24\jre\lib\resources.jar,c:\Program
Files\Java\jdk1.6.0_24\
jre\lib\rt.jar,c:\Program
Files\Java\jdk1.6.0_24\jre\lib\sunrsasign.jar,c:\Program
Files\Java\jdk1.6.0_24\jre\lib\jsse.j
ar,c:\Program Files\Java\jdk1.6.0_24\jre\lib\jce.jar,c:\Program
Files\Java\jdk1.6.0_24\jre\lib\charsets.jar,c:\Program F
iles\Java\jdk1.6.0_24\jre\lib\modules\jdk.boot.jar,c:\Program
Files\Java\jdk1.6.0_24\jre\classes,c:\Program Files\Java\j
dk1.6.0_24\jre\lib\ext\dnsns.jar,c:\Program
Files\Java\jdk1.6.0_24\jre\lib\ext\localedata.jar,c:\Program
Files\Java\jdk1
.6.0_24\jre\lib\ext\sunjce_provider.jar,.]
[loading java\lang\Object.class(java\lang:Object.class)]
[loading java\lang\String.class(java\lang:String.class)]
[checking ListArgs]
[loading java\lang\System.class(java\lang:System.class)]
[loading java\io\PrintStream.class(java\io:PrintStream.class)]
[loading
java\io\FilterOutputStream.class(java\io:FilterOutputStream.class)]
[loading java\io\OutputStream.class(java\io:OutputStream.class)]
[loading java\lang\StringBuilder.class(java\lang:StringBuilder.class)]
[loading
java\lang\AbstractStringBuilder.class(java\lang:AbstractStringBuilder.c
lass)]
[loading java\lang\CharSequence.class(java\lang:CharSequence.class)]
[loading java\lang\StringBuffer.class(java\lang:StringBuffer.class)]
[loading java\io\Serializable.class(java\io:Serializable.class)]
[loading java\lang\Comparable.class(java\lang:Comparable.class)]
[wrote ListArgs.class]
[total 533ms]
D:\ihrPfad>
```

5. Am Ende findet man im aktuellen Verzeichnis die neu dazugekommene *.class Datei. Das ist unsere Byte Code Datei. Diese ist nun mit Hilfe eines JRE ohne Weiteres ausführbar.

Wie starte ich Java Byte Code auf der Kommandozeile?

Die eben erzeugte Byte Code Datei (oder manchmal auch Class File genannt) lässt sich nun auf JEDEM Computer einer installierten JRE ausführen. Dazu muss man nur noch die JVM (java.exe) unter Angabe der Class Datei (*.class) starten. Dies macht man wie folgt:

1. Wir gehen davon aus das bereits ein JRE (oder JDK) auf dem Rechner installiert wurde.
2. Kommandozeile ist noch offen (cmd.exe)
3. Der Einfachheit halber wechseln Sie in das Verzeichnis wo die *.class gespeichert ist.
4. nun geben Sie ein:

   ```
   java ListArgs 111 222 333
   ```

   Achten Sie darauf dass wir hier nur den Namen der Java-Klasse und NICHT den Dateinamen angeben!! Die folgenden Zahlen sind nur Argumente für unser Programm damit es etwas zum Anzeigen hat.

5. Das Ergebnis sollte sein:

   ```
   Following Args been found:
   0 : 111
   1 : 222
   2 : 333

   D:\ihrPfad>
   ```

   Unser Programm kann nun also ausgeführt werden.

Will man sich die ganze Tipparbeit und langen Pfadnamen beim Programmaufruf sparen legt man sich am besten eine kleine Batchdatei an welche einem die Arbeit abnimmt. Das wird übrigens bei Java Anwendungen recht häufig gemacht.

Zum Schluss

Ich hoffe der Artikel war interessant und aufschlussreich. Wenn ich etwas falsch gemacht haben sollte oder Sie Verbesserungsvorschläge zum Thema haben freue ich mich über eine eMail und jedes Feedback.

Viel Erfolg beim programmieren mit Java!

— *Axel Werner 2011-09-08 21:10*

java, linux, windows, cli, programing