Skripte für Java im SWE-Labor

Java-Programme entwickeln kann man mit verschiedenen Editoren (z.B. TextPad, Notepad++, ...) oder mit speziellen Entwicklungsumgebungen (Eclipse, NetBeans, ...). Aber manchmal ist es nützlich, sich auf ganz einfache Werkzeuge zu beschränken, unter Windows z.B. auf eine *Eingabeaufforderung* (ein Fenster, in das man Kommandos eintippen kann) und `.cmd`-*Skripte* (die früher `.bat`-Skripte hießen).

Für die Windows-Rechner im SWE-Labor der Beuth Hochschule gilt: Normale Benutzer (im Gegensatz zu Administratoren) dürfen Umgebungsvariablen wie `PATH` und `CLASSPATH` nicht verändern. Mit Hilfe von Skripten kann man diese Einschränkung umgehen.

Grundlagen

Eine *Eingabeaufforderung* (oder Kommandozeile, engl. a Command Prompt) ist ein Fenster, in das man bestimmte Kommandos an das Betriebssystem eingeben kann und in das die Kommandos ihre Ausgabe ausgeben. Eine solche Eingabeaufforderung kann man z.B. wie folgt starten:

Linksklick auf den Startknopf, in die Eingabezeile **cmd.exe** eingeben, **Return**.

Oder: Linksklick auf den Startknopf, unter **Alle Programme** das Programm **Command Prompt** suchen und starten. Es befindet sich meistens im Ordner **Zubehör** oder im Ordner **Accessories**.

Gibt man in einer solchen Eingabeaufforderung z.B. folgendes Kommando (gefolgt von Return) ein

```
Z:\> calc
```

dann wird das Programm `calc.exe` gestartet, ein einfacher Taschenrechner.

Wenn Sie z.B.

```
Z:\> karlheinz
```

eingeben, erscheint sehr wahrscheinlich folgende Fehlermeldung:

```
Der Befehl "karlheinz" ist entweder falsch geschrieben oder
konnte nicht gefunden werden.
```

Frage: Wo hat das Betriebssystem nach einem Befehl `karlheinz` gesucht (und ihn nicht gefunden)?

Antwort: In allen Ordnern, die in der Umgebungsvariablen (engl. environment variable) namens `PATH` eingetragen sind.

Anmerkung: Es ist üblich, aber nicht nötig, Namen von Umgebungsvariablen mit *Großbuchstaben* darzustellen. Statt `PATH` kann man auch `path` oder `Path` etc. schreiben.

Mit dem folgenden Kommando (eingegeben in eine Eingabeaufforderung) kann man sich den Inhalt der Variablen `PATH` anzeigen lassen:

```
Z:\> echo %PATH%
```

Die Ausgabe kann z.B. wie folgt aussehen:

```
Z:\bin;C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows
\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;c:\Program Files(x86)\Jav
a\jdk1.8.0_20\bin;c:\msysgit;C:\Windows\System32\Windows System Resource Manager\bin
;C:\Program Files\jEdit;C:\Program Files (x86)\scala\\bin;C:\Program Files\jEdit510;
C:\Program Files (x86)\MySQL\MySQL Utilities\;C:\Program Files\TortoiseGit\bin;c:\pr
ogram files\common files\GTK\bin
```

Die `PATH`-Variable enthält eine Liste von *Pfadnamen von Ordnern*. Die einzelnen Pfadnamen sind durch je ein Semikolon `;` voneinander getrennt. Warum die Ausgabe des Kommandos `echo %PATH%` so schwer lesbar gestaltet wurde, ist mir nicht bekannt (Hatte Microsoft nicht genug Geld für ein besseres echo-Programm? Sollte der Inhalt der `PATH`-Variablen möglichst geheim bleiben?).

Wenn das Betriebssystem nach einem Befehl namens `karlheinz` sucht (in allen Ordnern, die in der Umgebungsvariablen `PATH` eingetragen sind), dann sucht es genauer gesagt nach einer *ausführbaren Datei* namens `karlheinz.exe` oder `karlheinz.cmd` oder `karlheinz.bat` oder ... . Welche

Erweiterungen (engl. extensions) eine ausführbare Datei haben darf, steht in der Umgebungsvariablen namens PATHEXT (wie path extensions). Die Ausgabe des Kommandos

```
Z:\> echo %PATHEXT%
```

kann z.B. so aussehen:

```
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
```

Die Umgebungsvariable PATH ist ein sehr wichtiges und zentrales Betriebsmittel des Betriebssystems Windows (und das Gleiche gilt auch für Unix-Betriebssysteme).

Unser Ziel: Wir möchten gern in der Lage sein, in einer Eingabeaufforderung Java-Programme zu kompilieren und auszuführen. Die Java-Programme sollen in verschiedenen Verzeichnissen stehen können. Wenn wir ein Programm bearbeiten, machen wir das betreffende Verzeichnis zu unserem Arbeitsverzeichnis. Die entsprechenden Kommandos zum Kompilieren und Ausführen wollen wir nicht jedes mal wieder (in die Eingabeaufforderung) eintippen, sondern wollen sie in `.cmd`-Skripte schreiben. Diese `.cmd`-Skripte sollen in einem Ordner `z:\bin` stehen. Diesen Ordner wollen wir in die PATH-Variable eintragen, damit wir die Skripte aufrufen können, unabhängig davon, in welchem Arbeitsverzeichnis wir uns gerade befinden. Da das dauerhafte Verändern der PATH-Variable im SWE-Labor nicht erlaubt ist, werden wir die Variable mit Hilfe eines `.cmd`-Skripts vorübergehend verändern.

Den Ordner z:\bin anlegen und füllen

Legen Sie (auf einem Windows-Rechner im SWE-Labor) einen Ordner `z:\bin` an.

Laden Sie das Archiv `Fuer-Z-bin.zip` herunter. Es enthält einen Ordner `Fuer-Z-bin` und darin vier Dateien (drei `.cmd`-Dateien und eine `.exe`-Datei). Kopieren Sie die vier Dateien in den Ordner `z:\bin`.

Eine der Dateien heißt `startCmd.cmd`. Legen Sie sich eine Verknüpfung zu dieser Datei auf Ihren Desktop (die Datei `startCmd.cmd` mit gedrückter rechter Maustaste auf den Desktop ziehen, die Maustaste loslassen und **Verknüpfungen erstellen** wählen). Ändern Sie den langen Namen der Verknüpfung zu `startCmd`.

Doppelklicken Sie auf die neue Verknüpfung. Eine Eingabeaufforderung sollte aufgehen.
Geben Sie folgendes Kommando ein:

```
Z.\> echu path
```

Die Ausgabe sollte etwa so aussehen:

```
-----------------------------------------------
Content of environment variable PATH
-----------------------------------------------
Z:\bin;
C:\ProgramData\Oracle\Java\javapath;
C:\Windows\system32;
C:\Windows;
C:\Windows\System32\Wbem;
C:\Windows\System32\WindowsPowerShell\v1.0\;
c:\Program Files (x86)\Java\jdk1.8.0_20\bin;
c:\msysgit;
C:\Windows\System32\Windows System Resource Manager\bin;
C:\Program Files\jEdit;
C:\Program Files (x86)\scala\\bin;
C:\Program Files\jEdit510;
C:\Program Files (x86)\MySQL\MySQL Utilities\;
C:\Program Files\TortoiseGit\bin;
c:\program files\common files\GTK\bin
```

Der Name "echu" soll an "echo für Umgebungsvariablen" erinnern.

Die PATH-Variable ist "eine zentrale Einrichtung des *Betriebssystems*". Die CLASSPATH-Variable ist "eine zentrale Einrichtung für den *Java-Ausführer*". Sie hat einen ähnlichen Inhalt wie die PATH-Varia-

ble: Eine Liste von Pfadnamen zu Ordnern oder zu `.jar`-Archiven (die ja etwas ähnliches wie Ordner sind, da sie mehrere Dateien enthalten können). Die einzelnen Pfadnamen sind (wie bei der PATH-Variablen) durch je ein Semikolon `;` voneinander getrennt. Etwas vereinfacht gesagt gilt: Immer wenn der Java-Ausführer (beim Kompilieren einer `.java`-Datei oder beim Ausführen einer `.class`-Datei) weitere Klassen benötigt, sucht er sie in den Ordnern und `.jar`-Archiven, die in der `CLASSPATH`-Variablen eingetragen sind.

Mit dem Kommando

```
Z:\> echu classpath
```

können Sie den Inhalt der `CLASSPATH`-Variablen (in einigermaßen lesbarer Form) anzeigen lassen. Das kann z.B. so aussehen:

```
-----------------------------------------------
Content of environment variable CLASSPATH
-----------------------------------------------
.;
..;
..\..;
c:\Program Files (x86)\java\junit.jar;
z:\klassen;
c:\Program Files (x86)\java\javamail-1.4.7\mail.jar
```

In diesem Beispiel enthält die `CLASSPATH`-Variable sechs Einträge, vier Ordner (oder: Verzeichnisse, engl. directories) und zwei `.jar`-Archive (`.;` bezeichnet das aktuelle Arbeitsverzeichnis, `..;` bezeichnet das Mutterverzeichnis von `.;` und `..\..;` bezeichnet das Mutterverzeichnis von `..;`).

Noch eine Ergänzung zur `PATH`-Variablen (siehe oben) : Beachten Sie, dass der Ordner `Z:\bin` in der `PATH`-Variablen steht: Das Skript `startCmd.cmd` hat ihn dort hineingeschrieben. Deshalb kann man alle Befehle, deren Dateien in diesem Ordner stehen, "von überall her" aufrufen. Die erweiterte `PATH`-Variable lebt nur solange, wie die vom Skript `startCmd.cmd` geöffnete Eingabeaufforderung läuft. Wenn man die Eingabeaufforderung schließt (mit dem Kommando `exit` oder durch einen Klick auf den Schließen-Knopf in der rechten oberen Ecke) gilt wieder die alte `PATH`-Variable (ohne `Z:\bin` darin).

Jetzt sollten Sie prüfen, ob man in der (vom Skript `startCmd.cmd` geöffneten) Eingabeaufforderung Java-Dateien kompilieren und Java-Programme ausführen lassen kann.

Navigieren Sie dazu mit der Eingabeaufforderung zu einem Ordner, in dem eine Java-Hallo-Datei namens `Hallo01.java` steht (oder ein ähnliches Java-Programm).

Geben Sie nacheinander die folgenden beiden Kommandos ein:

```
Z:\> c Hallo01.java
```

und dann

```
Z:\> a Hallo01
```

Dadurch sollte die Datei `Hallo01.java` kompiliert und die Datei `Hallo01.class` ausgeführt werden.

Die Skripte `startCmd.cmd`, `c.cmd` und `a.cmd` sind streng geheim. Sie dürfen Sie nur dann (z.B. mit dem Editor TextPad) öffnen, wenn Sie mindestens 18 Jahre alt sind und sich wirklich für Informatik interessieren.

Eine Eingabeaufforderung verschönern

Von Hause aus ist eine Eingabeaufforderung ein relativ kleines Fenster mit weißer Schrift auf schwarzem Hintergrund. Einige Eigenschaften dieses Fensters kann man relativ leicht verändern.

Ein Links-Doppelklick auf die Verknüpfung `startCmd.cmd`. Eine Eingabeaufforderung (klein, schwarzer Hintergrund etc.) öffnet sich.

Ein Rechtsklick in die linke obere Ecke des Fensters, auf das Icon links neben dem Namen **startCmd**. Wählen Sie in dem sich öffnenden Menü ganz unten **Eigenschaften**. Dadurch sollte ein Fenster aufgehen mit dem Titel **Eigenschaften von "startCmd"** und vier Reitern (**Optionen**, **Schriftart**, **Layout**, **Farbe**) darin.

Wählen Sie den Reiter **Farben**.
Wählen Sie für **Fensterhintergrund** die Farbe weiß und dann für **Fenstertext** die Farbe schwarz (das wird Ihren Augen gut tun).

Wählen Sie den Reiter **Layout**.
Stellen Sie ein: **Fensterpuffergröße**, **Breite**: 100, **Höhe** 300.
Stellen Sie ein: **Fenstergröße**, **Breite**: 100, **Höhe**: 40.

Wählen Sie den Reiter **Schriftart**.
Wählen Sie im Fensterchen unter **Größe** die Schriftgröße 12x16.

Klicken Sie (im **Eigenschaften von "startCmd"**-Fenster) unten auf **OK**.

Die Eingabeaufforderung sollte ihre Erscheinung jetzt stark verändert haben. Wenn Ihnen irgendeine Eigenschaft nicht gefällt, dann wiederholen Sie den hier beschriebenen Einstellungsvorgang und probieren Sie andere Einstellungen aus (z.B Magenta für den Fensterhintergrund und Hellgrün für den Fenstertext? Fragen Sie in Zweifelsfällen Ihren Arzt oder Apotheker, aber keinen Optiker, denn der profitiert davon, wenn Ihre Augen schlechter werden :-).

Wenn Sie (durch einen Links-Doppelklick auf die Verknüpfung `startCmd`) eine Eingabeaufforderung öffnen, ist immer ein ganz bestimmter Ordner Ihr aktuelles Arbeitsverzeichnis. Wenn dieser Ordner weit weg liegt von den Ordnern, in denen Sie gewöhnlich Java-Programme entwickeln (so dass mit vielen `cd`-Kommandos zu "Ihren Ordnern" hin-navigieren müssen), dann können Sie einen günstigeren Ordner einstelle wie folgt:

Rechts-Klick auf die Verknüpfung `startCmd` und **Eigenschaften** wählen. Ein Fenster mit dem Titel **Eigenschaften von startCmd** und vielen Reitern öffnet sich. Wählen Sie den Reiter **Verknüpfung**. Rechts neben **Ausführen in:** können Sie einen beliebigen (existierenden) Ordner eintragen. In Zukunft wird dieser Ordner Ihr aktuelles Arbeitsverzeichnis sein, wenn Sie über die Verknüpfung `startCmd` eine Eingabeaufforderung öffnen.

Möglicherweise lohnt es sich, diese Einstellung häufig zu ändern (immer wenn Sie damit beginnen, ein neues Java-Programm in einem neuen Ordner zu entwickeln). Oder Sie legen sich verschiedene Verknüpfungen `startCmd-1`, `startCmd-2`, ... an, jede mit einem anderen **Ausführen-in**-Ordner.

Auf Ihrem eigenen Windows-Rechner

Auf Ihrem eigenen Rechner können Sie die Dateien aus dem Archiv `Fuer-Z-bin.zip` in irgend einen Ordner kopieren und den Pfad, der zu diesem Ordner führt, in die `PATH`-Variable eintragen. Das Skript `startCmd.cmd` benötigen Sie gar nicht. Im Skript `c.cmd` müssen Sie den Pfadnamen `z:\klassen` durch einen anderen Pfadnamen (z.B. `c:\klassen`) ersetzen und den Ordner `c:\klassen` erzeugen.