Einführung in die IOTools, Installation und Anwendung

Inhalt

---

1 Tastatureingaben in Java

Java ist eine komplexe Sprache - eine Sprache, in der man wahrscheinlich fast alles programmieren kann, wenn man nur weiß wie.

Um den großen Umfang an Funktionalität und Plattformunabhängigkeit zur Verfügung zu stellen (welchen die Sprache nun einmal hat), haben die Entwickler viele Dinge abstrahiert. Diese wurden daraufhin in so allgemeiner Form implementiert, dass insbesondere Anfänger große Schwierigkeiten haben, sie zu benutzen. Leider ist die Eingabe eine davon.

Java erhält Eingaben von Daten über sogenannte Ströme (engl.: *streams*). Man kann sich diese Ströme am besten wie einen altmodischen Nachrichtenticker vorstellen. Auf einem schmalen Streifen kommen Nachrichten an, Zeichen für Zeichen aneinandergereiht. Welche Daten
und Informationen auf diesen Streifen stehen, ist dem Gerät dabei egal - es handelt sich nur um eine Ansammlung von Zeichen. Es obliegt dem Leser, die Streifen an der richtigen Stelle abzureißen und die Daten auf dem Streifen zu interpretieren.

So oder so ähnlich können wir uns auch die Ströme in Java veranschaulichen. Ein Strom besteht aus nichts weiter als einer Ansammlung von Bits, die durch das Programm in irgendeiner Form interpretiert werden müssen. Hierbei kann die Quelle dieser Zeichen verschieden sein: eine Datei auf der Festplatte, ein Dokument aus dem Internet, oder eben die Eingabe von der Tastatur. Die Methodenaufrufe, welche dem Benutzer zur Verfügung stehen, sind in allen Fällen gleich.

Haben wir es jedoch geschafft dem Computer klarzumachen, dass wir als Eingabestrom die Tastatur verwenden wollen, so sind wir damit noch lange nicht am Ziel. Das Beste, was wir dem PC mit den vordefinierten Methoden nämlich entlocken können, ist eine Kette von Zeichen - ein *String*. Wir wollen im allgemeinen jedoch keine *Strings*, sondern *int*-Werte, *double*-Zahlen oder eventuell einzelne Zeichen. Zwar gibt es Möglichkeiten, aus unserem String diese zu extrahieren; hierfür brauchen wir jedoch meistens Wissen, das über das eines Anfängers hinausgeht. Um dem Abhilfe zu schaffen, wurden die *IOTools* geschrieben.

---

2 Installation

Möchten Sie die Klasse *IOTools*
in Ihren Programmen unter Windows 95/98/Me, Windows NT/2000/(XP), Linux/Unix oder Mac-OS bzw. in speziellen Entwicklungsumgebungen verwenden, so gehen Sie wie in den jeweils zutreffenden nachfolgenden Unterpunkten beschrieben vor.

2.1 Grundlegendes

Wir gehen davon aus, dass Sie eine JDK-Version neuer (also größer) als 1.2 verwenden.

Besorgen Sie sich die Datei *Prog1Tools.zip*
und speichern Sie sie auf Ihrer Festplatte. Schreiben Sie sich auch diesen Pfad auf. Wir gehen einmal davon aus, dass die Position der zip-Datei

```
c:\programme\java\other\Prog1Tools.zip
```

für Windows und

```
~/java/other/Prog1Tools.zip
```

für Linux/Unix ist.

Das weitere Vorgehen bei der Installation ist abhängig vom verwendeten Betriebssystem und von eventuell verwendeten Editoren bzw. Entwicklungsumgebungen für Java.

2.2 Weiteres Vorgehen für Windows 95/98/Me

Öffnen Sie nun die Datei *autoexec.bat* im Verzeichnis *c:* mit einem Editor. Fügen Sie die Zeile

```
set CLASSPATH=.;c:\programme\java\other\Prog1Tools.zip;
```

ein. Beachten Sie unbedingt den Punkt (.) nach dem Gleichheitszeichen!

Nun (genauer gesagt, nach dem Ausführen obiger Befehls-Zeile beim nächsten Neustart Ihres Rechners) können Sie in einem Konsolenfenster wie gewohnt mit *javac* kompilieren, ohne auf spezielle Parameter achten zu müssen.

2.3 Weiteres Vorgehen für Windows NT/2000/XP

Öffnen Sie das Fenster *Systemsteuerung*, dann den Menüpunkt *System* und wählen dort den Punkt *Erweitert* aus. Dort haben Sie die Möglichkeit unter *Umgebungsvariablen...* Benutzervariablen/Systemvariablen zu setzen. Bitte beachten Sie, dass für das Setzen von Systemvariablen ein Administratorpasswort benötigt wird.
Fügen Sie nun folgende Variablen als Benutzer- oder als Systemvariablen ein (je nachdem gilt die Einstellung nur für den jeweiligen Benutzer oder für alle Benutzer dieses Rechners!):

Fügen Sie die Variable

```
Name der Variablen:    CLASSPATH

Wert der Variablen:    .;c:\programme\java\other\Prog1Tools.zip
```

ein. Beachten Sie unbedingt den Punkt (.) zu Beginn der Zeile *Wert der Variablen*!

Nun können Sie in einem Konsolenfenster wie gewohnt mit *javac* kompilieren, ohne auf spezielle Parameter achten zu müssen.

2.4 Weiteres Vorgehen für Linux/Unix

Editieren Sie die Datei *~/.profile* und tragen Sie dort ein:

```
CLASSPATH=.:~/java/other/Prog1Tools.zip:$CLASSPATH;
export CLASSPATH
```

Anmerkung: Wenn Sie als Shell die C-Shell (csh oder tcsh) verwenden, müssen in die Datei *~/.cshrc* die Zeilen

```
set CLASSPATH = ( $CLASSPATH . )
set CLASSPATH = ( $CLASSPATH ~/java/other/Prog1Tools.zip )
```

eingefügt werden.

Nun (genauer gesagt, nach dem Ausführen obiger Befehls-Zeile beim nächsten Login) können Sie in einem neuen

Konsolenfenster wie gewohnt mit *javac* kompilieren, ohne auf spezielle Parameter achten zu müssen.

2.5 Weiteres Vorgehen für Mac-OS

Auf dem Macintosh unter **Mac-OS 9.x** genügt es, wenn Sie die Datei *Prog1Tools.zip* ins Verzeichnis

```
Systemordner:Systemerweiterungen:MRJ Libraries:MRJClasses:
```

legen.

Unter **Mac-OS X**, das auf Unix bzw. Linux basiert, kann wie unter 2.4 beschrieben vorgegangen werden.

2.6 Weiteres Vorgehen für spezielle Java-Editoren oder Java-Entwicklungsumgebungen (Die IOTools in Eclipse, JOE, JBuilder etc.)

Wollen Sie Ihre Java-Programme, die Methoden aus den IOTools verwenden, direkt innerhalb von speziellen Editoren für Java (z. B. JOE) oder von speziellen Entwicklungsumgebungen für Java (z. B. Eclipse oder JBuilder) compilieren und ausführen, so muss auch innerhalb der Optionen dieser Editoren bzw. Entwicklungsumgebungen der Classpath so eingestellt werden, dass beim Compilieren und beim Ausführen die IOTools-Klassen gefunden werden. Normalerweise verwenden auch diese Entwicklungsumgebungen die im Betriebssystem (gemäß 2.2 bis 2.5) eingestellten Werte der Variable CLASSPATH, so dass prinzipiell nichts zusätzliches eingestellt werden muss! Werden die IOTools beim Compilieren bzw. Starten der Programme innerhalb der verwendeten Entwicklungsumgebung jedoch nicht gefunden, so ist wie nachfolgend beschrieben vorzugehen.

In ***Eclipse*** gehen Sie wie folgt vor:

> Im Menüpunkt *Window* dessen Unterpunkt *Preferences* wählen. Im *Preferences Fenster* den Eintrag *Java* und dessen Unter-Eintrag *Installed JREs* aufklappen (z. B. durch Doppelklick mit der linken Maustatste). Danach mit der rechten Maustaste einmal auf den angehakten *JRE-Eintrag* klicken. Mit der linken Maustaste auf den Button *Edit* drücken. Es erscheint ein Kontext-Menü. Dort den Button *Add External JARs* klicken und die Datei *Prog1Tools.zip*, also
>
> ```
> c:\programme\java\other\Prog1Tools.zip
> ```
>
> auf Ihrem Rechner suchen und selektieren. Diese erscheint jetzt als Untereintrag der *JRE system libraries*. Danach zweimal den OK-Button klicken.

Im ***JOE*** (Java Oriented Editor) können sie diese Einstellungen wie folgt verändern:

> Im Menüpunkt *Optionen* und dessen Unterpunkt *Einstellungen* finden Sie unter *Java JDK* den Eintrag *Zusätzlicher Klassenpfad*, dort müssen Sie den Pfad zu den IOTools auf Ihrem Rechner eintragen, also
>
> ```
> .;c:\programme\java\other\Prog1Tools.zip
> ```
>
> Beachten Sie auch hier unbedingt den Punkt (.) zu Beginn des Eintrags.

Im ***JBuilder*** können sie diese Einstellungen wie folgt verändern:

> Im Menüpunkt *Tools* den Eintrag *JDKs konfigurieren* wählen. Dort müssen dann den Knopf *Hinzufügen* drücken, um ein neues Paket
> (nämlich die Prog1Tools) hinzuzufügen. die Datei *Prog1Tools.zip*, also
>
> ```
> c:\programme\java\other\Prog1Tools.zip
> ```
>
> auf Ihrem Rechner suchen und selektieren. *OK* klicken, das neue Paket erscheint in der Liste der Klassen-Libraries (letzter Eintrag).
> Danach nochmals *OK* klicken.

In ***Forte für Java*** können sie diese Einstellungen wie folgt verändern:

> Im Menüpunkt *Project* dessen Unterpunkt *Settings* wählen. Im *Project Settings Fenster* den Eintrag *Filesystem Settings*
> aufklappen (z. B. durch Doppelklick mit der linken Maustatste). Danach mit der rechten Maustaste einmal

auf *Filesystem Settings* klicken. Es erscheint ein Kontext-Menü. Dort den Punkt *Mount JAR* auswählen die Datei *Prog1Tools.zip*, also

```
c:\programme\java\other\Prog1Tools.zip
```

auf Ihrem Rechner suchen und selektieren. Diese erscheint jetzt als Untereintrag der *Filesystem Settings*.

Für andere Editoren/Entwicklungsumgebungen müssen Sie analoge Einstellungen vornehmen.

---

3 Anwendung

Wollen Sie die Klasse *IOTools*
in Ihrem Programm verwenden, so müssen Sie in die erste Zeile ihres Programms nur noch

```
import Prog1Tools.IOTools;
```

einfügen.

Folgende Methoden sind dadurch unter anderem bereitgestellt:

- Die Methode *readInteger* bzw. *readInt* liest eine Zahl vom Typ *int* von der Tastatur ein und gibt diese als Ergebnis zurück. Um beispielsweise zwei ganze Zahlen von der Tastatur einzulesen und in den Variablen *a* und *b* zu sichern, genügt folgendes Programmstück:

  ```
  int a = IOTools.readInteger();
  int b = IOTools.readInteger();
  ```

- Die Methode *readDouble* liest eine Zahl vom Typ *double* ein. Obiges Beispiel würde also für *double*-Zahlen wie folgt aussehen:

  ```
  double a = IOTools.readDouble();
  double b = IOTools.readDouble();
  ```

- Die Methode *readLong* liest eine Zahl vom Typ *long* ein. Die Methoden *ReadShort* und *ReadFloat* tun dies für die Datentypen *short* und *float*.
- Die Methode *readLine* liest eine ganze Textzeile (abgeschlossen durch den Druck auf die Eingabetaste).
- Die Methode *readString*
  liest ein einzelnes "Textwort" von der Tastatur. Ein solches Textwort besteht aus einem *String*, der weder durch Leer- noch Tabulator- bzw. Zeilenende-Zeichen auseinandergerissen ist. Geben wir beispielsweise die Zeile

  ```
  Dies ist eine schoene Zeile
  ```

  ein und rufen den Befehl *readString*
  auf, so liefert er lediglich Dies als Ergebnis. Um an das nächste Wort zu gelangen, muss die Methode erneut aufgerufen werden.
- Die Methode *readChar*
  liest ein einzelnes Zeichen, welches nicht gleich dem Leerzeichen, Zeilenendezeichen oder dem Tabulatorzeichen ist. Die Methode basiert hierbei auf der *readString*-Methode, das heißt, es werden Textworte eingelesen und in ihre einzelnen Komponenten aufgespalten.
- Das Programmstück

  ```
  IOTools.readChar();
  char a=IOTools.readChar();
  int b=IOTools.readInteger();
  ```

  liefert bei der Eingabe *abc123 456* also *a='b'* und *b=456*, da die Ziffern *123* noch zum ersten Textwort gehören.
- Die Methode *readBoolean*
  liest einen boolschen Wert ein. Hierbei ist auf Groß- und Kleinschreibung zu achten; die Eingabe *True* kodiert beispielsweise keinen Wert vom Typ *boolean*. Es muss vielmehr *true* heißen. Wie wir in obigen Beispielen gesehen haben, können auch mehr als eine einzulesene Information pro Zeile eingegeben werden (man muss sie lediglich durch Leerzeilen trennen). Hierbei muss man natürlich auf die Reihenfolge der Eingaben achten. Der Befehl *readInteger* wird bei der Eingabe

  ```
  Ich gebe jetzt einmal 13 ein.
  ```

als Ergebnis den Wert 13 zurückgeben, da dies die erste gültige Ganzzahl ist. Die zuvor stehenden Textworte werden verworfen.

Letzte Änderung: 3.1.2007 (D. Ratz)