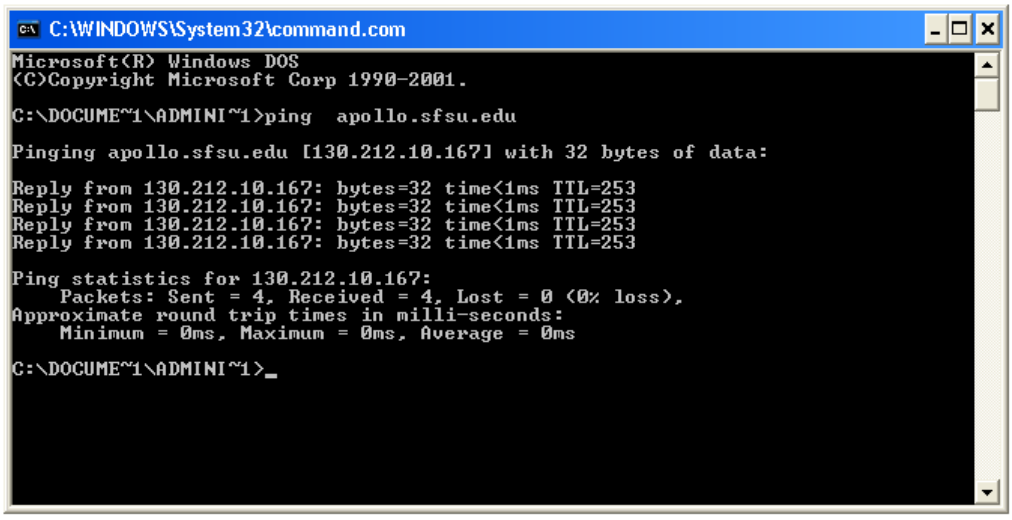
Java programmieren: Konsolen-Programme

Es war einmal vor langer Zeit in einer weit entfernten Galaxis... ok, das ist etwas übertrieben. In den Anfängen der Personal Computer in den 1980er sahen Computer noch etwa so aus:

Diese ersten Personal Computer kannten nur monochrome (einfarbige) Bildschirme, die zudem nur Text ausgeben konnten. Computer mit grafischen Benutzeroberflächen kamen erst später auf: erfunden an Xerox Parc, popularisiert von Apple und schliesslich von Microsoft weit verbreitet.

In den folgenden Java-Programmier-Aufgaben machen wir einen Ausflug in die Welt der sogenannten **Konsolen**, auch **Kommandozeilen** (engl. Shell) genannt, siehe auch **http://de.wikipedia.org/wiki/Kommandozeile**. Konsole bedeutet, die Interaktion mit dem Computer geschieht ausschliesslich mit der Tastatur und dem Bildschirm, und die Anzeige ist beschränkt auf Text. In Windows zum Beispiel kann die Kommandozeile gestartet werden, in dem im Startmenü bei „Ausführen“ der Befehl „cmd“ eingeben wird. Man erhält dann eine einfache Kommandozeile:



Auch wenn die meisten Anwender nie mit der Konsole arbeiten: Sie spielen auch heute noch eine zentrale Rolle, vor allem beim Betrieb von Servern. In einer typischen sogenannten Server-Farm hat es hunderte, tausende und sogar zehntausende von Computern, die aus der Ferne betreut werden. Das geschieht typischerweise über die Konsole, vor allem bei Servern, auf denen ein **Unix-basiertes Betriebssystem** läuft. Mächtig werden die Unix-Shells (Kommandozeilen bei Unix werden meist mit Shell bezeichnet) dadurch, dass es sog. **Shell Script Programming Languages** gibt. Mit einer Shell-Programmiersprache wie etwa Bash können auch Programme geschrieben werden. Der folgende Einzeiler zum Beispiel lädt 31 Dateien von einem Server runter:

```
for i in `seq -f "%02.f" 1 31`; do wget https://unser.server.com/logs/access.log.201012$i.gz --http-user=benutzername --http-password=passwort --no-check-certificate; done
```

Für Hintergrundsinformationen und eine Einführung zu Shell Skripting siehe auch

http://de.wikipedia.org/wiki/Unix-Shell
http://de.wikipedia.org/wiki/Bourne-again_shell
http://www.selflinux.org/selflinux/html/shellprogrammierung.html

Einfache Java-Konsolen-Programme: Ausgabe

Betrachten wir zunächst das berühmte „Hallo Welt“-Programm, das für eine Programmiersprache zeigen soll, wie „einfach“ eine Textausgabe mit dieser Sprache funktioniert. Hier das Programm für Java, mit Konsolen-Ausgabe:

```
public class HalloWelt {
	public static void main(String[] args) {
		System.out.println("Hallo, Welt!");
	}
}
```

- Wie immer muss der Klassenname **HalloWelt** dem Dateinamen entsprechen (HalloWelt.java).
- Das Hauptprogramm, der Einstiegspunkt in das Programm, ist die Methode **main**. Diese Methode muss exakt wie im Beispiel oben definiert werden.
- Da die Methode main **static** ist – ohne darauf näher einzugehen – müssen auch alle unsere eigenen Methoden im gleichen Programm als static definiert werden. Beispiele folgen unten.

Die eigentliche Ausgabe erfolgt mit dem Aufruf der Methode **System.out.println ("schreibe Text in Zeile")**. Dieser Methode wird eine Zeichenkette („Hallo, Welt!“) übergeben. Diese wird auf die Konsole geschrieben. Anschliessend wird der unsichtbare Cursor in die nächste Zeile verschoben. Unsichtbarer Cursor? Man kann sich die Ausgabe auf die Konsole wie folgt vorstellen. Die Konsole ist eine Tabelle von gleichgrossen Feldern, in die Zeichen geschrieben werden können. Die Tabelle selbst ist dabei unsichtbar. Zu Beginn ist die Tabelle leer, und der unsichtbare Cursor seht auf dem ersten Feld in Leserichtung, dargestellt durch einen vertikalen Strich |.

Nach der Ausgabe von „Hallo, Welt“ mit dem Aufruf von System.out.println sieht die Konsole wie folgt aus:

H	a	l	l	o	,		W	e	l	t	!					

Soll der Cursor nach der Textausgabe nicht in die nächste Zeile rücken, kann die Methode **System.out.print ("schreibe Text ohne anschliessenden Zeilenumbruch")** verwendet werden. Diese Methode schreibt den Text ebenfalls auf die Konsole, aber ohne den Cursor auf die nächste Zeile vorzurücken. Nach System.out.print("Hallo, Welt!") würde die Konsole daher wie folgt aussehen:

H	a	l	l	o	,		W	e	l	t	!					

Eine Herausforderung von Konsolen-Programmen ist die Darstellung von Grafiken nur mit Zeichen in dem fixen Raster der Konsole. Es gibt sogar Leute, die damit einen künstlerischen Anspruch einzulösen versuchen: Unter dem Begriff ASCII Art (ASCII ist die Auswahl an Zeichen die ursprünglich gemäss amerikanischem Standard zur Verfügung stand) versteht man Zeichnungen auf der Konsole, siehe auch http://de.wikipedia.org/wiki/ASCII-Art.

Wir betrachten hier ein etwas einfacheres Beispiel. Ein Programm soll eine diagonale Linie ausgeben. Damit die Linie besser sichtbar wird, soll die Linie selbst mit Leerzeichen dargestellt werden und die rechteckige Umgebung mit Sternchen gefüllt werden:

```
 ****
* ***
** **
*** *
****
```

Das Programm dazu sieht wie folgt aus:

```
public static void main(String[] args) throws IOException {
      final BufferedReader konsolenEingabe = new BufferedReader(
                  new InputStreamReader(System.in));
      final int eingabe =
                  Integer.parseInt(konsolenEingabe.readLine());

      for (int i = 0; i < eingabe; i++) {
            wiederholeZeichen(i, '*');
            System.out.print(' ');
            wiederholeZeichen(eingabe - i - 1, '*');
            System.out.println();
      }
}

static void wiederholeZeichen(final int anzahl, char c) {
      for (int i = 0; i < anzahl; i++) {
            System.out.print(c);
      }
}
```

- Zunächst wird mit *Integer.parseInt(konsolenEingabe.readLine());* eine Zahl von der Konsole eingelesen; die Zeile davor macht die Eingabe möglich, ist aber hässlich und wird von uns nicht näher betrachtet. Der Benutzer muss die Eingabe mit der Enter-Taste abschliessen. Gibt er keine Zahl ein, erhält er eine Java-Fehlermeldung angezeigt.

- Die for-Schleife im Hauptprogramm zeichnet die Diagonale von links oben nach rechts unten. Dazu wird die Methode wiederholeZeichen verwendet, die ein Zeichen eine Anzahl mal ausgibt.
- Wichtige **Unterscheidung**: Eine Variable vom Typ **String** ist eine Zeichenkette wird wie im Beispiel "Hallo, Welt" in doppelten Anführungszeichen geschrieben. Man kann einen String lesen, ihn aber nicht verändern. Eine Variable vom Typ **char** ist ein einzelnes Zeichen und wird wie im obigen Beispiel '*' in einfachen Anführungszeichen geschrieben.

Verständnisfragen

- Was passiert, wenn Sie eine leere Eingabe tätigen? Wenn Sie Text eingeben?
- Wo wird in dem Programm die Ausgabe auf einer neuen Zeile gestartet?
- Variieren Sie die Zeichen, die für die Ausgabe verwendet werden, sowohl das Leerzeichen als auch das Sternchen – finden Sie eine bessere Darstellung für die Diagonale.
- Änderung Sie das Programm, so dass die Diagonale von links unten nach rechts oben geht.
- Was passiert, wenn Sie in wiederholeZeichen zum Beispiel *3 * anzahl* Zeichen ausgeben?

Einfache Java-Konsolen-Programme: Textanalyse

Häufig müssen Texte analysiert werden. Betrachten wir ein einfaches Programm, das in einer Eingabe die Anzahl Gross- und Kleinbuchstaben sowie die Anzahl Nicht-Buchstaben zählt:

```
public class GrossbuchstabenKleinbuchstabenZaehlen {

	public static void main(String[] args) throws IOException {
		final BufferedReader konsolenEingabe = new BufferedReader(
				new InputStreamReader(System.in));
		final String eingabe = konsolenEingabe.readLine();

		int anzahlGrossbuchstaben = 0;
		int anzahlKleinbuchstaben = 0;
		int anzahlNichtBuchstaben = 0;

		for (int i = 0; i < eingabe.length(); i++) {
			char c = eingabe.charAt(i);
			if (Character.isUpperCase(c)) {
				anzahlGrossbuchstaben++;
			} else if (Character.isLowerCase(c)) {
				anzahlKleinbuchstaben++;
			} else {
				anzahlNichtBuchstaben++;
			}
		}

		System.out.println("Anzahl Grossbuchstaben: " +
				anzahlGrossbuchstaben);
		System.out.println("Anzahl Kleinbuchstaben: " +
				anzahlKleinbuchstaben);
		System.out.println("Anzahl Nicht-Buchstaben: " +
				anzahlNichtBuchstaben);
	}

}
```

- Mit String *eingabe = konsolenEingabe.readLine();* wird eine Zeichenkette eingelesen; mit der Taste Enter wird die Eingabe beendet. Das Zeichen für Enter wird dabei nicht in eingabe gespeichert.
- Die Methodenaufruf *eingabe.length();* liefert die Länge der String-Variablen eingabe zurück.
- Die Anweisung *char c = eingabe.charAt(i);* speichert das i-te Zeichen aus der String-Variable eingabe in der Variablen c. Mit charAt kann ein String wie ein Array vom Typ char der Länge eingabe.length() betrachtet werden.
- Die Aufrufe der Hilfsmethoden *Character.isUpperCase(c)* und *Character.isLowerCase(c)* geben true zurück, wenn die Eingabe c ein Gross- bzw. Kleinbuchstabe ist.

Verständnisfragen

- Was passiert, wenn Sie eine leere Eingabe tätigen? Wenn Sie Text eingeben?
- Begründen Sie mit einem Beispiel, wozu es das erste „else“ im Programm braucht: Was würde ohne dieses else falsch gezählt?
- Erweitern Sie das Programm, so dass die Häufigkeit von Gross- und Klein und Nicht-Buchstaben in Prozent ausgegeben wird. Achten Sie darauf, dass Ihr Programm keinen Fehler produziert, falls der Benutzer eine leere Eingabe tätigt.

Das folgende Programm ermittelt die durchschnittliche Länge der eingegebenen Wörter:

```
public class StringEingabeDurchschnittlicheLaenge {

	public static void main(String[] args) throws IOException {
		final BufferedReader konsolenEingabe = new BufferedReader(
				new InputStreamReader(System.in));

		int summeLaengen = 0;
		int anzahlEingaben = 0;

		String eingabe = konsolenEingabe.readLine();
		while (!"".equals(eingabe)) {
			anzahlEingaben++;
			summeLaengen = summeLaengen + eingabe.length();
			eingabe = konsolenEingabe.readLine();
		}

		System.out.println("Anzahl Eingaben: " + anzahlEingaben);

		int durchschnittlicheLaenge = summeLaengen / anzahlEingaben;
		System.out.println("Durchschnittliche Länge der Eingaben: "
				+ durchschnittlicheLaenge);
	}

}
```

- Für eine String-Variable gibt die Methode equals true zurück, falls die beiden Strings exakt gleich sind (inkl. Gross- und Kleinschreibung). *"".equals(eingabe)* ist dann true, wenn eine leere Eingabe erfolgte.

- Die Anweisung für die Eingabe kommt zwei Mal vor: Einmal für das erste Mal, ausserhalb der while-Schleife, und dann wieder als Abschluss der Anweisungen in der while-Schleife.

Verständnisfragen

- Was passiert, wenn Ihre erste Eingabe leer ist, und warum? Wie müssen Sie das Programm anpassen, damit es auch mit einer einzigen leeren Eingabe klarkommt?
- Könnten Sie die Bedingung in der while-Schleife auch anders schreiben als mit der Prüfung auf Gleichheit?
- Ändern Sie das Programm so ab, dass es auch die minimale und maximale Eingabelänge ausgibt.

Einfache Java-Konsolen-Programme: Texte verändern

Betrachten wir ein Programm, das eine ganz primitive Art der „Kompression“ vornimmt: Bei Wörtern werden immer höchstens die ersten vier Buchstaben ausgegeben:

```
Rafael Nadal war im Viertelfinal des Australian Open – auch wegen der Ver-
letzung – mit 4:6, 2:6, 3:6 gegen seinen Landsmann David Ferrer ausgeschie-
den.
Rafa Nada war im Vier des Aust Open – auch wege der Verl – mit 4:6, 2:6,
3:6 gege sein Land Davi Ferr ausg.
```

Die Ausgabe ist doch immer noch fast verständlich:-) Das Programm dazu:

```
public class ErsteVierBuchstaben {

	public static void main(String[] args) throws IOException {
		final BufferedReader konsolenEingabe = new BufferedReader(
				new InputStreamReader(System.in));
		final String eingabe = konsolenEingabe.readLine();

		int anzahlBuchstabenImWort = 0;
		for (int i = 0; i < eingabe.length(); i++) {
			if (Character.isLetter(eingabe.charAt(i))) {
				if (anzahlBuchstabenImWort < 4) {
					System.out.print(eingabe.charAt(i));
					anzahlBuchstabenImWort++;
				}
			} else {
				System.out.print(eingabe.charAt(i));
				anzahlBuchstabenImWort = 0;
			}

		}
	}

}
```

- Die Methode Character.isLetter(c) gibt true zurück, wenn die Eingabe c ein Buchstabe ist, sonst false (wenn c zum Beispiel eine Zahl oder ein Sonderzeichen ist).

Verständnisfragen

- Warum braucht es an zwei Orten im Programm den Aufruf von System.out.print(...)?
- Ist die Ausgabe des Programmes auch dann korrekt, wenn vor dem ersten Wort zum Beispiel Leerzeichen eingegeben werden? Begründen Sie Ihre Antwort.
- Ergänzen Sie das Programm so, dass auch bei Zahlen wie 123456 und bei kombinierten Eingaben wie passwort1234 höchstens die ersten vier Zeichen ausgegeben werden. Sie brauchen dazu die Methode Character.isDigit(c).

Zusammenfassung für einfache Java-Konsolen-Programme

- Wichtige **Unterscheidung**: Eine Variable vom Typ **String** ist eine Zeichenkette wird wie im Beispiel "Hallo, Welt" in doppelten Anführungszeichen geschrieben. Man kann einen String lesen, ihn aber nicht verändern. Eine Variable vom Typ **char** ist ein einzelnes Zeichen und wird wie im obigen Beispiel '*' in einfachen Anführungszeichen geschrieben:
  String s = "Hallo, Welt!";
  char c = 'r';
- Zwei **Strings aneinanderhängen** mit + ergibt eine neue Variable vom Typ String:
  String name = konsolenEingabe.readLine();
  System.out.println("Hallo, " + name);
- String-**Eingabe in Ganzzahlen** umwandeln:
  final int eingabe = Integer.parseInt(konsolenEingabe.readLine());
- Die Methodenaufruf *eingabe.length();* liefert die **Länge der String**-Variablen eingabe zurück.
- Die Anweisung *char c = eingabe.charAt(i);* speichert **das i-te Zeichen aus der String-Variable** eingabe in der Variablen c. Mit charAt kann ein String wie ein Array vom Typ char der Länge eingabe.length() betrachtet werden.
- Die Aufrufe der Hilfsmethoden *Character.isUpperCase(c)* und *Character.isLowerCase(c)* geben true zurück, wenn die Eingabe c ein **Gross- bzw. Kleinbuchstabe** ist. *Character.toLowerCase(c)* bzw. *Character.toUpperCase(c)* **wandelt Buchstaben in Klein- bzw. Grossbuchstaben** um.
- Die Methode *Character.isLetter(c)* gibt true zurück, wenn die **Eingabe c ein Buchstabe** ist, sonst false (wenn c zum Beispiel eine Zahl oder ein Sonderzeichen ist).
- Die Methode *Character.isDigit(c)* gibt true zurück, wenn die **Eingabe c eine Ziffer** ist, sonst false (wenn c zum Beispiel ein Buchstabe oder ein Sonderzeichen ist).

Quellen

http://openbook.galileocomputing.de/javainsel8/javainsel_04_001.htm