gds2 Gottlieb-Daimler-Schule 2
Technisches Schulzentrum Sindelfingen
mit Abteilung Akademie für Datenverarbeitung

Textuelle Eingabe von der Tastatur und Ausgabe auf dem Bildschirm bei Konsolprogrammen

1 Ausgabe von textuellen Daten

Alle Daten, die auf dem Bildschirm in einer Textkonsole (Eingabeaufforderung, Kommandozeile) angezeigt werden sollen, müssen zunächst in eine Zeichenkette (String) umgewandelt werden. Möchte man eine Zeichenkette ausgeben, ist also keine Konvertierung notwendig.

Bsp.:
```
System.out.println("Hello World");
```

Beim Start des Programms wird automatisch das Objekt `System.out` erzeugt, das sozusagen eine Verbindung zur Textkonsole herstellt. Mit der Methode `println` kann dann eine Ausgabe auf die Konsole geschickt werden. Der Text, der als Parameter mitgegeben wird, erscheint auf dem Bildschirm. Da `println` für „printLine“ steht, wird zusätzlich der „Cursor“ an den Anfang der nächsten Zeile gesetzt.

Statt einem String-Literal kann auch ein Variableninhalt ausgegeben werden:

```
String zeichenkette = "Hello"

System.out.println(zeichenkette);  // Ausgabe: "Hello" + "<CR><LF>"[1]
System.out.println(zeichenkette + " World"); // Ausgabe: "Hello World"[2]
```

2 Ausgabe von numerischen Daten

Wenn numerische Daten in der Konsole ausgegeben werden sollen, so ist Konvertierung notwendig. Die interne binäre Darstellung von Zahlen muss in eine entsprechende Zeichenfolge (ebenfalls intern binär) umgewandelt werden, damit die Zahl auf der Konsole dargestellt werden kann.

```
System.out.println(768);  // Ausgabe: "768"
int zahl = 768;
System.out.println(zahl); // Ausgabe: "768"
System.out.println("Zahl=" + zahl); // Ausgabe: "Zahl=768"
```

Die Methode `println` führt die Konvertierung automatisch durch. Es sind dabei allerding ein paar Besonderheiten zu beachten:

```
System.out.println("Zahl=" + zahl + 2); // Ausgabe: "Zahl=7682"
```

Beide Plus Operatoren werden hier nicht als arithmetische Additionsoperatoren interpretiert, sondern als String-Konkatenationsoperatoren. Mit zusätzlichen Klammern:

```
System.out.println("Zahl=" + (zahl + 2)); // Ausgabe: "Zahl=770"
```

[1] <CR><LF> steht für Carriage Return Line Feed: Windows verwendet für einen Zeilenvorschub diese zwei unsichtbaren Zeichen.
[2] Der Zeilenvorschub erfolgt bei println in jedem Fall immer und wird ab jetzt nicht mehr ausdrücklich angegeben.

3 Ausgabe mit expliziter Formatierung

Sowohl in Konsolprogrammen als auch in Programmen mit grafischer Oberfläche (GUI[3]) kann es notwendig sein, dass eine Ausgabestring formatiert ausgegeben wird.

Für die formatierte Ausgabe wird statt der Methode `println` die Methode `printf` (print formatted) verwendet.

Beispiel: Ausgabe einer Fließkommazahl

```
System.out.println("Pi = " + 3.141592653589793 + "ist eine Konstante");
```

`Println` erzeugt die Ausgabe: `Pi = 3.141592653589793 ist eine Konstante`

Println wandelt die interne Darstellung der Fließkommazahl in eine entsprechende Dezimalzahl um. Dabei wird eine Standardformatierung angewendet. Alle signifikanten Stellen (Vor- und Nachkommastellen) werden ausgegeben. Als Dezimaltrennzeichen wird der Punkt verwendet (wie im Englischen üblich).

```
System.out.printf("Pi = %14.10f ist eine Konstante", 3.141592653589793);
```

`Printf` erzeugt die Ausgabe: `Pi =   3,1415926536 ist eine Konstante`

Printf wandelt die interne Darstellung der Fließkommazahl in eine Dezimalzahl um, wobei das Format der Dezimalzahl von dem „Format-Spezifizierer“ `"%14.10f"` gesteuert wird. Der erste Parameter der printf-Methode ist ein String. Er enthält Zeichenketten, die wortwörtlich am Bildschirm angezeigt werden sollen (im Beispiel `Pi =  ist eine Konstante`). Der „Format-Spezifizierer“ `"%14.10f"` hat zwei Funktionen: er steht als Platzhalter für einen Wert, der an seiner Stelle ausgegeben werden soll. Dieser Wert muss als weiterer Parameter der printf-Methode mitgegeben werden (im Beispiel: `3.141592653589793)`. Die zweite Funktion besteht darin, dass Format der Ausgabe zu spezifizieren (im Beispiel: „reserviere 14 Zeichen für die Ausgabe der Zahl, runde die Zahl auf 10 Nachkommastellen und gebe die Zahl rechtsbündig innerhalb der 14 Zeichen aus)

Folgende Format-Spezifizierer sind definiert:

%d	Ganzzahl als Dezimalzahl ausgeben
%h	Ganzzahl als Hexadezimalzahl ausgeben
%f	Fließkommazahl (double, float, ...) ausgeben
%e	Fließkommazahl (double, float, ...) in wissenschaftlicher Notation ausgeben
%b	boolean Wert ausgeben
%s	String wortwörtlich ausgeben
%t	Datum / Zeit
%%	Ersatzdarstellung wenn tatsächlich wortwörtlich eine Prozentzeichen ausgegen werden soll
%n	Gibt <CR><LF> aus und erzeugt damit einen Zeilenvorschub

[3] GUI: Graphical User Interface (grafische Benutzeroberfläche)

Beispiele:

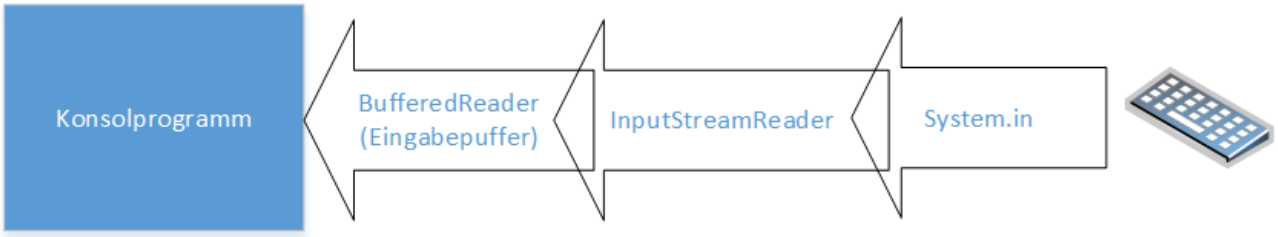
```
int i = 123;
System.out.printf("|%d|   |%d|%n" ,      i, -i );   // |123|   |-123|
System.out.printf("|%5d| |%5d|%n" ,      i, -i );   // |  123| | -123|
System.out.printf("|%-5d| |%-5d|%n" ,    i, -i );   // |123  | |-123 |
System.out.printf("|%+-5d| |%+-5d|%n" , i, -i );   // |+123 | |-123 |
System.out.printf("|%05d| |%05d|%n",  i, -i );     // |00123| |-0123|
System.out.printf("|%X|  |%x|%n",        i,  i );   // |7B|  |7b|
System.out.printf("|%04x| |%#x|%n%n",   i,  i );   // |007b| |0x7b|

double d = 12.345;
System.out.printf("|%f| |%f|%n" ,          d, -d ); // |12,345000| |-12,345000|
System.out.printf("|%.2f| |%.2f|%n" ,     d, -d ); // |12,35| |-12,35|
System.out.printf("|%10.2f| |%10.2f|%n" , d, -d ); // |     12,35| |    -12,35|
System.out.printf("|%010.2f| |%010.2f|%n",d, -d ); // |0000012,35| |-000012,35|
d = d * 1000;
System.out.printf("|%,10.2f| |%,10.2f|%n" d, -d ); // | 12.345,00| |-12.345,00|

String s = "Feierabend";
System.out.printf("%n|%s|%n", s );                  // |Feierabend|
System.out.printf("|%20s|%n", s );                  // |          Feierabend|
System.out.printf("|%-20s|%n", s );                 // |Feierabend          |
System.out.printf("|%7s|%n", s );                   // |Feierabend|
System.out.printf("|%.5s|%n", s );                  // |Feier|
System.out.printf("|%20.5s|%n", s );                // |               Feier|
```

4 Gepufferte Tastatureingabe bei Konsolprogrammen

4.1 Auf dem harten Weg: mit Basisklassen



Das Einlesen von Daten von der Tastatur ist tatsächlich eine komplexere Angelegenheit, als man zunächst vermutet, wie man im folgenden Quelltext sieht:

```
 1 public static void main(String[] args) throws IOException
 2 {
 3       BufferedReader br
 4           = new BufferedReader(new InputStreamReader(System.in));
 5
 6       String eingabe;
 7       int ganzzahl;
 8
 9       System.out.println("Geben Sie eine Zeile ein:");
10
11       // Lesen einer Textzeile inklusive <CR><LF> und Speichern
12       // in der Variablen eingabe (ohne <CR><LF>)
13       eingabe = br.readLine();
14       System.out.println("Eingebene Zeichenkette: " + eingabe);
15
16       // Konvertierung des Textes in eine interne Ganzzahl (int)
17       ganzzahl = Integer.parseInt(eingabe);
18
19       System.out.printf("Zahl = %10d\n", ganzzahl);
20 }
```

1. Herstellen einer Verbindung vom Programm zur Tastatur: Zeile 3 und 4
   Liest ein Programm Daten von der Tastatur, so geschieht dies immer indirekt über das Betriebssystem. (Alle Hardwarezugriffe müssen über das Betriebssystem ausgeführt werden.)
   Beim Start eines Konsolprogramms wird automatisch das Objekt `System.in` erzeugt. Dies erlaubt dem Programm einzelne Bytes von der Tastatur zu lesen.
   Da man aber normalerweise Zeichen lesen möchte und sich ein Zeichen aus mehreren Bytes zusammensetzen kann, benötigt man noch ein Objekt der Klasse `InputStreamReader`.
   Meistens möchte man, dass eine Eingabe erst dann weiter verarbeitet wird, wenn der Benutzer die <Enter> Taste drückt. Deshalb müssen alle Zeichen bis auf das <Enter> zunächst zwischengepuffert werden. Dafür benötigt man ein Objekt der Klasse `BufferedReader` (hier hat es den Bezeichner `br`).

2. Der Befehl in Zeile 13 sorgt dafür, dass das Programm anhält, und auf die Benutzereingabe von der Tastatur wartet. Angenommen der Benutzer gibt 4711 ein. Erst wenn er <Enter> drückt, wird die Zeichenkette "4711" aus dem Puffer in die String Variable `eingabe` kopiert und das Programm läuft weiter.

3. Zeile 17:
   Jede Eingabe, die von dem Tastaturpuffer kommt, ist zunächst eine Zeichenkette. Möchte man die Zeichenkette "4711" als Zahl interpretieren und in einer Ganzzahlvariablen speichern, so muss diese Zeichenkette erst noch in das interne binäre Ganzzahlformat konvertiert werden. Dies kann man mit einem Befehl wie in Zeile 17 erreichen. Hat der Benutzer allerdings statt Ziffern Buchstaben eingegeben, so würde der Befehl in Zeile 17 zu einem Programmabsturz führen, da man Buchstaben nicht in eine Zahl konvertieren kann.

4.2 Bequemeres Einlesen mit Hilfe einer vordefinierten Klasse

Wie wir bereits kennengelernt haben, ist eine Grundprinzip der Programmierung die Modularisierung. In der OOP kann man mit Hilfe von Klassen und deren Methoden Module erstellen, die komplexere Vorgänge lösen und dem Anwender die entsprechende Funktionalität zur Verfügung stellen.

Wir verwenden in Zukunft die selbstgeschriebene Klasse Eingabe mit ihren Methoden um das Einlesen von der Tastatur sehr einfach zu bewerkstelligen.

```
 1 import input.Eingabe;
 2
 3 public class EingabeIO
 4 {
 5       public static void main(String[] args)
 6       {
 7             String eingabe;
 8             int ganzzahl;
 9
10             eingabe = Eingabe.getString("Geben Sie eine Zeile ein:");
11
12             System.out.println("Eingebene Zeichenkette: " + eingabe);
13
14             ganzzahl = Eingabe.getInt("Geben Sie eine Zahl ein:");
15
16             System.out.printf("Zahl = %10d\n", ganzzahl);
17
18       }
19 }
```

Um die Klasse `Eingabe` mit ihren Methoden (wie z.B. `getString` oder `getInt`) verwenden zu können, sind drei Schritte notwendig:

1. Laden Sie sich die Java Archivdatei `input.jar` von Moodle in das Verzeichnis auf Laufwerk H:\ wo sich auch der Eclipse Workspace befindet. (jar steht für Java Archive)
2. Rufen Sie in Eclipse das Menü: `Project`→`Preferences` auf. Wählen Sie links den Punkt „`Java Build Path`“. Rechts wählen Sie dann „`Add External Jars`“; navigieren zu der Datei `input.jar` und binden diese in den „Build Path“ ein. Damit sind alle Klassen aus dieser Archiv Datei in Eclipse bekannt.

3. Fügen Sie oben in der Quelltextdatei die Zeile `import input.Eingabe;` hinzu. Dann können Sie die Eingabemethoden verwenden. Damit man in Eclipse zu den einzelnen Methoden auch noch eine kurze Beschreibung sichtbar machen kann, laden wir außerdem noch die Datei input.zip von Moodle herunter, entpacken diese und kopieren sie in das Verzeichnis
   `H:\..\workspaceVerzeichnis\Projektverzeichnis\src`