Scanner – Einlesen mittels Systemklassen

Lange Zeit mussten Java-Programmierer sich damit abfinden, daß es keine vorgefertigten API-Systemklassen für das einfache Einlesen von Wert- und Objektdatentypen in einem Konsolenfenster gab; deshalb entstanden einige eigene Ersatzklassen, wie z.B. IO-Tools, siehe Kapitel 5a). Ab Java 5 gibt es Systemfunktionen, mit deren Hilfe das plattformunabhängige Einlesen von der Tastatur möglich ist. Sie befinden sich im API-Paket ***java.util.Scanner***.

Die Java-Dokumentation bietet eine ausführliche Beschreibung der zahlreichen Scanner-Methoden:

http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html?is-external=true

An den folgenden einfachen Programmbeispielen wird die Benutzung der Scanner-Methoden deutlich.

```
// Ganzzahlen (int)

import java.util.Scanner;

class ScannerTest {

      public static void main(String [] args) {

            Scanner s = new Scanner (System.in);

            System.out.println( "Eingabe einer Ganzzahl: ");

            int i = s.nextInt();

            System.out.println( "Ausgabe: "+i );
      }
}
```

```
// Gleitkommazahlen (double)

import java.util.Scanner;

class ScannerTest {

      public static void main(String [] args) {

            Scanner s = new Scanner (System.in);

            System.out.println("Eingabe Gleitkommazahl: ");

            double d = s.nextDouble();

            System.out.println( "Ausgabe: "+ d);
      }
}
```

Achtung: Das Format des Dezimaltrenners ist hier im Unterschied zu vielen anderen Systemfunktionen lokalisiert, d.h. an das interne Landesschema (locale) angepaßt - also: Nicht PUNKT, sondern KOMMA.

```
// Einlesen von Zeichenketten:

import java.util.Scanner;

class ScannerTest {

      public static void main(String [] args) {

            Scanner s = new Scanner (System.in);

            System.out.println( "Bitte Eingabe: ");

            String t = s.nextLine();

            System.out.println( "Ausgabe: "+ t);
      }
}
```

Alle elementaren Datentypen haben solche Input-Methoden, die den jeweiligen Datentyp als Rückgabewert haben:

`nextBoolean, nextByte, nextDouble, nextFloat, nextInt, nextLong, nextShort usw. …`

– außer CHAR (who knows why ;-)

Der Aufruf aller dieser Eingabemethoden ist blockierend, das heisst, daß der Programmablauf an der Stelle des Aufrufs stehen bleibt und wartet, bis eine Eingabe (die immer mit einem **RETURN** abgeschlossen werden muss) erfolgt.

Zu beachten ist, daß bei mehrmalig aufeinanderfolgendem Aufruf von **`next( )`** (ohne Datentypangabe) der Eingabepuffer nicht automatisch geleert wird. Um herauszufinden, ob der Puffer leer ist, kann die Methode **`hasNext( )`** benutzt werden.