Technische Universität Braunschweig

IAS | INSTITUTE FOR APPLICATION SECURITY

Programmieren Vorkurs

Input/Output, If, Bedingungen

Thole Goesmann, 10.10.2018

Über mich

Thole Goesmann

- Studiere Mathematik und Informatik
- HiWi am Institut für Anwendungssicherheit
- gewähltes Mitglied im FGR Informatik
- Mail: `t.goesmann@tu-braunschweig.de`

Gliederung

- Input/Output
- if Statement
- Conditions

Input/Output

print/println

- `System.out.print(s);`

  Gibt den übergebenen `String` s aus.

- `System.out.println(s);`

  Gibt den übergebenen `String` s aus und beginnt eine neue Zeile.

Input/Output

print/println

- `System.out.print(s);`

  Gibt den übergebenen `String` s aus.

- `System.out.println(s);`

  Gibt den übergebenen `String` s aus und beginnt eine neue Zeile.

- `System.out.printf(String s)`

  Für komplexe Formatierung, wird hier aber nicht verwendet.

Input/Output

Escape Character

- Es gibt Zeichen die in Java schon eine spezielle Bedeutung haben, diese kann man nicht einfach in seinen Quellcode schreiben. Diese werden stattdessen durch eine Zeichenfolge dargestellt, die immer mit einem bestimmten Zeichen dem Escape Character beginnt.

- `\"` Anführungszeichen
- `\n` Zeilenumbruch
- `\t` Tab
- `\\` Backslash

Input/Output

Übergabeparameter

- Beim Programmstart können nach dem Programmnamen noch Parameter übergeben werden.

```
user@foo~$ java Bar param1 param2 ...
```

Input/Output

Übergabeparameter

- Beim Programmstart können nach dem Programmnamen noch Parameter übergeben werden.

```
user@foo~$ java Bar param1 param2 ...
```

- Diese sind in der `main()` im `String` Array Parameter zu finden.

```
public static void main(String[] args) {
  // args = {param1, param2, ...};
  System.out.println(args.length);
}
```

Input/Output

Scanner

- Zur Eingabe während der Laufzeit kann man die Scanner-Klasse verwenden.
- Diese muss zunächst importiert werden und es muss ein neuer Scanner erstellt werden.
- Mittels `next()` und `nextLine()` kann das nächste Wort oder die nächste Zeile eingelesen werden.
- Mit `nextInt()` kann das nächste Wort als `int` eingelesen werden.

Input/Output

Scanner

- Zur Eingabe während der Laufzeit kann man die Scanner-Klasse verwenden.
- Diese muss zunächst importiert werden und es muss ein neuer Scanner erstellt werden.
- Mittels `next()` und `nextLine()` kann das nächste Wort oder die nächste Zeile eingelesen werden.
- Mit `nextInt()` kann das nächste Wort als `int` eingelesen werden.
- Mit `Integer.parseInt()` kann ein `String` zu einem `int` konvertiert werden.

Input/Output

Beispiel

```
import java.util.Scanner;
public class Foo {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    String input = Scanner.nextLine();
    System.out.println(input);
    int zahl = Integer.parseInt(Scanner.next());
    System.out.println(input);
    int zahl2 = Scanner.nextInt();
    System.out.println(zahl + zahl2);
  }
}
```

if Statement

if

- Was tun wenn Anweisungen nur ausgeführt werden sollen, falls eine Bedingung erfüllt ist?
- Z.B. Wenn es eine Eingabe gibt soll diese bearbeitet werden.

if Statement

if

- Was tun wenn Anweisungen nur ausgeführt werden sollen, falls eine Bedingung erfüllt ist?
- Z.B. Wenn es eine Eingabe gibt soll diese bearbeitet werden.
- In Java gibt es hierfür `if` Statements.

```
if (condition) {
    ...
}
```

if Statement

else

- Nach einem `if` Statement kann ein `else` Statement stehen.
- Dieses wird ausgeführt, falls die Bedingung im `if` Statement nicht erfüllt ist.

```
if (condition) {
    ...
} else {
    ...
}
```

if Statement

else if

- Nach einem `if` Statement kann auch ein `else if` Statement stehen.
- Dieses besitzt auch eine Bedingung .
- Es wird ausgeführt, wenn seine Bedingung erfüllt ist und die des vorhergehenden `if` Statements nicht erfüllt ist.

if Statement

else if

- Nach einem `if` Statement kann auch ein `else if` Statement stehen.
- Dieses besitzt auch eine Bedingung .
- Es wird ausgeführt, wenn seine Bedingung erfüllt ist und die des vorhergehenden `if` Statements nicht erfüllt ist.

```
if (condition) {
    ...
} else if (condition2) {
    ...
}
```

if Statement

- Nach einem `else if` Statements können weitere `else if` Statements stehen.
- Nach einem `else if` Statement kann auch ein abschließendes `else` Statement stehen.

if Statement

- Nach einem `else if` Statements können weitere `else if` Statements stehen.
- Nach einem `else if` Statement kann auch ein abschließendes `else` Statement stehen.

```
if (condition) {
    ...
} else if (condition2) {
    ...
} else if (condition3) {
    ...
} else {
    ...
}
```

Conditions

- Bedingungen sind Aussagen, die zum Zeitpunkt der Auswertung eindeutig wahr oder falsch sein müssen.
- Bedingungen müssen für den Compiler verständlich formuliert sein.

Conditions

- Bedingungen sind Aussagen, die zum Zeitpunkt der Auswertung eindeutig wahr oder falsch sein müssen.
- Bedingungen müssen für den Compiler verständlich formuliert sein.
- In Java sind Bedingungen `booleans`.

```
boolean rain = true;
if (rain) {
 System.out.println("It's raining.");
}
```

Conditions

Vergleichen von Zahlen

- Operatoren zum Vergleichen von Zahlen geben ebenfalls `booleans` zurück. D.h. sind sie auch Bedingungen.

Conditions

Vergleichen von Zahlen

- Operatoren zum Vergleichen von Zahlen geben ebenfalls `booleans` zurück. D.h. sind sie auch Bedingungen.

- $=$ == (= ist schon für die Zuweisung reserviert.)
- $<$ <
- $>$ >
- $\leqslant$ <=
- $\geqslant$ >=
- $\neq$!=

Conditions

Vergleichen von anderen Datentypen

- `boolean` können ebenfalls mit „==“ auf Gleichheit getestet werden.
- Zum Vergleichen von `Strings` und anderen Objekten gibt es die `equals()` Funktion.

Conditions

Vergleichen von anderen Datentypen

- `boolean` können ebenfalls mit „==“ auf Gleichheit getestet werden.
- Zum Vergleichen von `Strings` und anderen Objekten gibt es die `equals()` Funktion.

```
String foo = "bar";
boolean bool = true;
if (foo.equals("bar") == bool) {
        System.out.println("foobar");
}
```

Conditions

Logische Operatoren

- Bedingungen/`booleans` können miteinander verknüpft werden.
- z.B. Es regnet und es scheint die Sonne.
- Bedingungen können außerdem negiert (verneint) werden.
- z.B. Es regnet nicht.
- Wenn nicht klar ist in welcher Reihenfolge Operatoren angewendet werden sollen, muss diese (formal) mit () angegeben werden.

Conditions

Operatorenübersicht

Name	Formal	Java	wahr, falls
and	$\vee$	&&	beide Bedingungen wahr sind.
or	$\wedge$	||	eine der Bedingungen wahr ist.
exclusive or	xor	^	genau eine der Bedingungen wahr ist.
not	$\neg$	!	die Bedingung falsch ist.

Conditions

Wahrheitstabellen

- Geben die Werte von Bedingungen abhängig von `booleans` an.
- Eine 0 entspricht hier falsch und eine 1 entspricht wahr.

A	B	A && B	A || B	A ^ B	!A	(A && B) || !A
0	0	0	0	0	1	1
0	1	0	1	1	1	1
1	0	0	1	1	0	0
1	1	1	1	0	0	1

Fragen?

- Raum für Notizen

Linux-Installparty

Mittwoch, 10. Oktober 16 Uhr in IZ 161 (gegenüber FG-Raum)

- Freies Betriebssystem (kostenlos & Quellcode verfügbar)
- Nützlich in Studium & jeglichen Lebenssituationen
- Installation neben vorhandenem Betriebssystem möglich
- **Wir begleiten euch bei der Installation und den ersten Schritten!**

Bitte führt vorher ein **Backup eurer Daten** durch, um einen evtl. Verlust auszuschließen. Siehe z.B.
`https://www.heise.de/tipps-tricks/Backup-erstellen-mit-Windows-10-3858841.html`