Grundlagen der Programmierung

Vorlesungsskript zum Sommersemester 2020

3. Vorlesung (4. Mai 2020)

TECHNISCHE UNIVERSITÄT DARMSTADT

Dr. Jin Gerlach
Christian Olt
Fachgebiet Wirtschaftsinformatik | Software & Digital Business
Fachbereich Rechts- und Wirtschaftswissenschaften
Technische Universität Darmstadt

WIRTSCHAFTS INFORMATIK

True oder false?

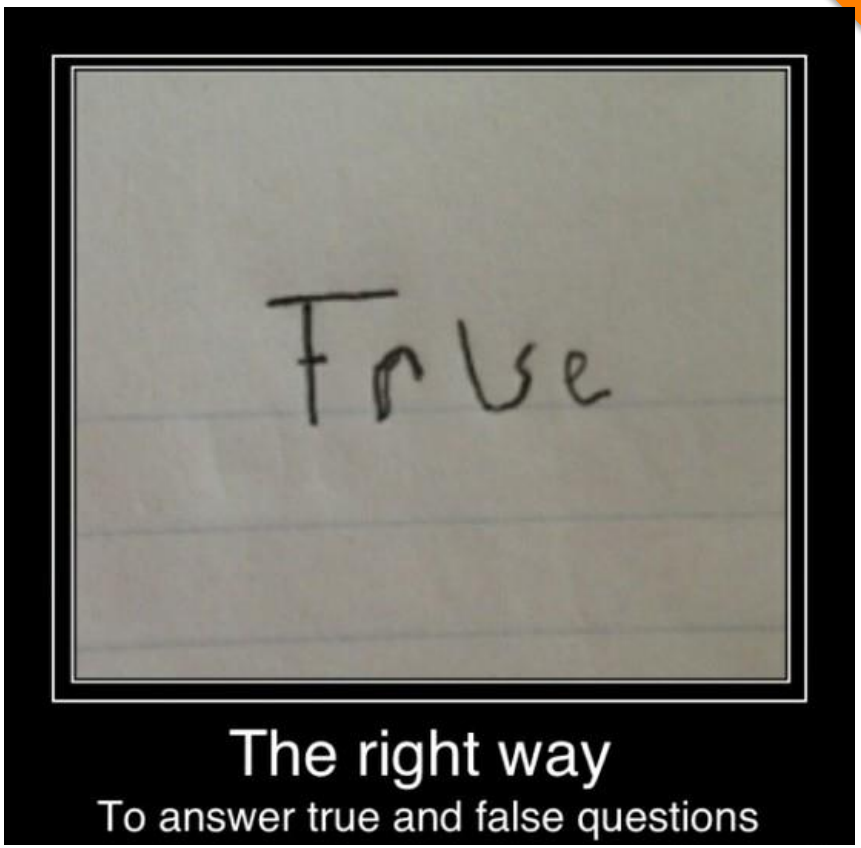
http://pingo.upb.de
#9456

1. Wenn ein Kontrollfluss innerhalb einer Aktivität verzweigt wird, müssen von dieser Aktivität mindestens zwei Kontrollflüsse ausgehen.

2. Bei der Deklaration einer Variablen muss ein Wert angegeben werden, mit dem die Variable initialisiert wird.

3. Variablen vom Datentyp „boolean“ können ganzzahlige Werte zwischen -128 und +127 annehmen.

4. Einzeilige Kommentare beginnen in Java mit einem „//“.

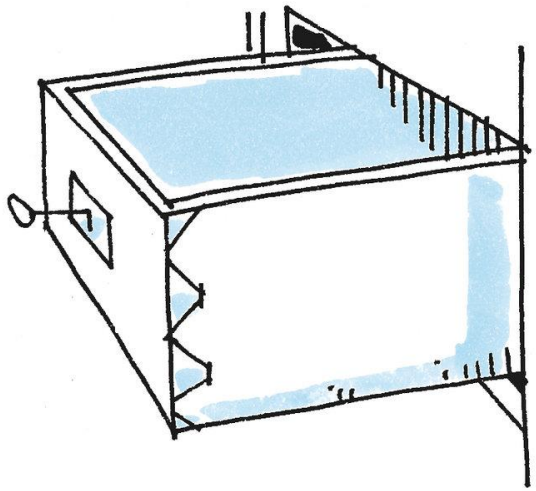


Modellierungsregeln für Aktivitätsdiagramme

- Es werden keine Entscheidungen bzw. Verzweigungen des Kontrollflusses innerhalb einer Aktivität modelliert
- Innerhalb einer Aktivität werden nur Handlungen des gleichen Typs zusammengefasst (Hinweis: Wir unterscheiden die drei Typen von Aktivitäten: Eingaben, Ausgaben, Operationen)
- Jedes Diagramm „startet“ mit genau einem Startknoten und „endet“ mit genau einem Endknoten
- Es „entspringen“ niemals mehr als ein Kontrollfluss aus dem Startknoten oder einer Aktivität
- Für jede Verzweigung sind auf den ausgehenden Kanten alle möglichen Fälle anzugeben (Hinweis: „sonst“ verwenden)

Variablen und Arten von Datentypen

- **Variablen** sind Behälter für genau einen Wert und es gilt:
  - Variablen haben einen **Datentyp** (z. B. `int`)
  - Variablen haben einen **Bezeichner** (z. B. `x` oder `besteVariableDerWelt`)
  - Variablen haben einen **Wert** (z. B. `42`)



- Wir unterscheiden **zwei Arten von Datentypen**:
  - **Einfache** Datentypen:
    - Wahrheitswert (`boolean`) → true, false
    - Einzelzeichen (`char`) → z. B.: `'a'`, `'b'`, `'1'`, `'2'`, `'%'`, `'_'`
    - Numerische Datentypen
      - Ganzzahlige Datentypen (`byte`, `short`, `int`, `long`) → z. B. 123
      - Gleitkommatypen (`float`, `double`) → z. B. 1.25
  - **Strukturierte** Datentypen (auch: Referenz-Datentypen)

Deklaration und Initialisierung von Variablen

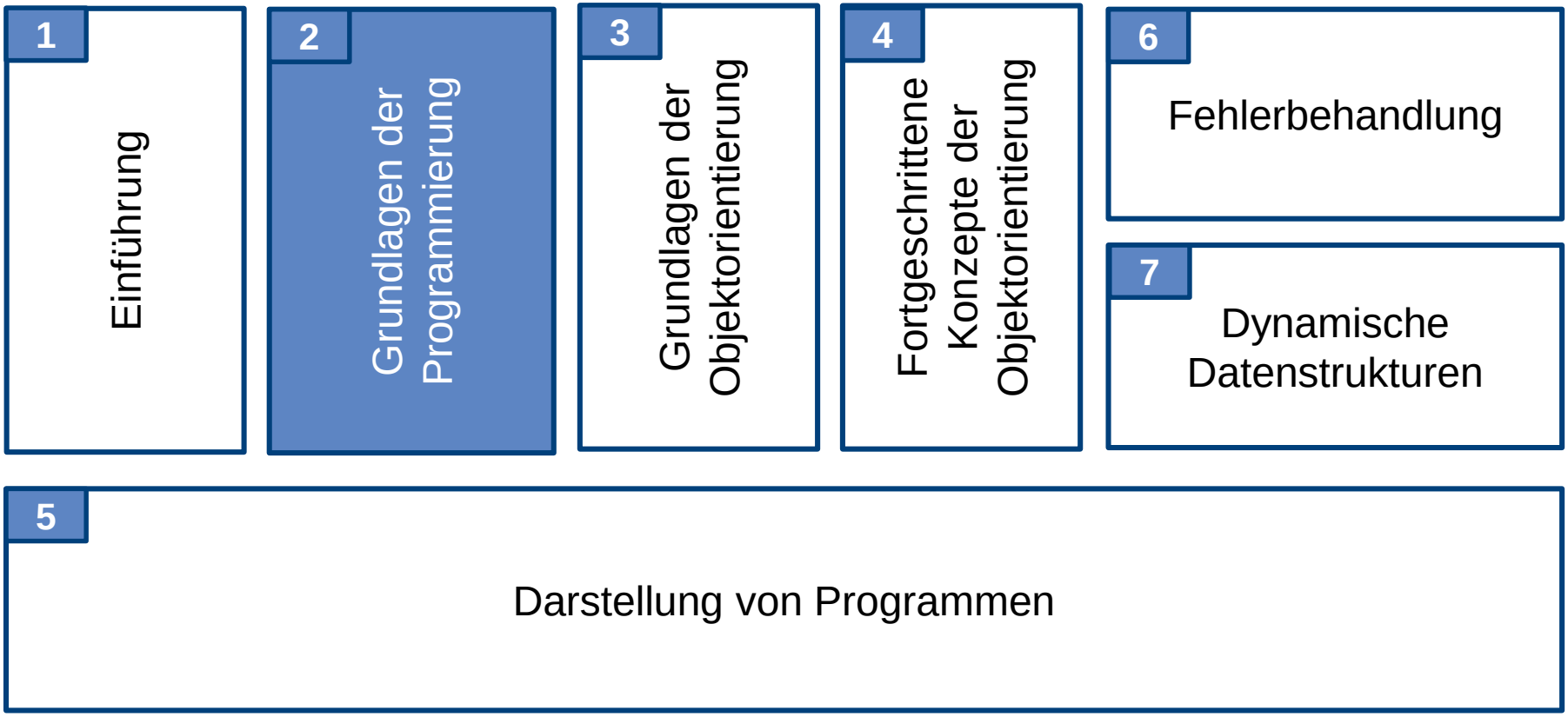
- Variablen müssen vor der ersten Verwendung im Programm **deklariert** werden, dazu gibt man den **Datentyp** an und vergibt einen **Bezeichner**, optional kann die Variable auch **initialisiert** werden, d. h. sie bekommt schon einen ersten Wert zugewiesen. Beispiele für die **Deklaration** einer Variable:

```
int x; // Variablendeklaration ohne Initialisierung
int y = 42; // Variablendeklaration mit Initialisierung
int z = y + x + 1; // Variablendeklaration mit Initialisierung
```

- Variablen können im Programm **gelesen** und **geschrieben** werden. Ein Lesen der Variablen ist notwendig, wenn sie auf der rechten Seite einer Zuweisung verwendet werden

Thematische Übersicht

TECHNISCHE UNIVERSITÄT DARMSTADT



WIRTSCHAFTS INFORMATIK

Wiederholung

Kapitel 2: Grundlagen der Programmierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aktivitätsdiagramme der Unified Modeling Language (UML)
- Deklaration, Initialisierung und Nutzung von Variablen in Java
- Wichtige Elemente der Programmiersprache Java

Lernziele:

- Den Aufbau von Aktivitätsdiagrammen kennen und gegebene Algorithmen durch Aktivitätsdiagramme modellieren können
- Variablen deklarieren, initialisieren und verwenden können
- Wesentliche Elemente eines Java Programms kennen und verstehen

Kommentare

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kommentare dienen der Erläuterung von Quellcode, werden beim Kompilieren verworfen und haben somit keine Auswirkungen auf die Programmlogik
- Zwei wesentliche Arten von Kommentaren:
  - **Einzeilige Kommentare** erstrecken sich bis zum Zeilenende, z. B.:

    ```
    System.out.println("text"); // Kommentar
    ```
  - **Mehrzeilige Kommentare** erstrecken sich über mehrere Zeilen, z. B.:

    ```
    System.out.println("text"); /* erste Zeile
                                    zweite Zeile */
    ```

Schlüsselwörter

TECHNISCHE UNIVERSITÄT DARMSTADT

- Schlüsselwörter haben eine vordefinierte symbolische Bedeutung und können nicht als Bezeichner verwendet werden:

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	extends	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	package	private	protected	public
return	short	static	strictfp	super
switch	synchronized	this	throw	throws
transient	try	void	volatile	while

Bezeichner

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Bezeichner** (Namen) benennen Variablen, Methoden, Klassen und Schnittstellen, etc. und ermöglichen es so, die entsprechenden Bausteine anschließend im Programm identifizieren und zu referenzieren
- Hinweis: Bei Bezeichnern und anderen Elementen unterscheidet Java strikt zwischen **Groß- und Kleinschreibung**!
- Folgende Regeln sollen bei **Vergeben von Bezeichnern** beachtet werden:
  - Erstes Zeichen: A-Z, a-z, _, $
  - Weitere Zeichen: A-Z, a-z, _, $, 0-9
  - Es dürfen keine Schlüsselwörter verwendet werden
  - „true“, „false“, „null“ sind verboten
  - Drei weitere Bezeichner sind Namen vordefinierter Klassen und dürfen somit nicht verwendet werden: „Object“, „String“, „System“

Beispiel: Ungültige Bezeichner

TECHNISCHE UNIVERSITÄT DARMSTADT

Ungültige Bezeichner	Grund
2und2macht4	Das erste Symbol muss ein Java-Buchstabe sein und keine Ziffer.
hose gewaschen	Leerzeichen sind in Bezeichnern nicht erlaubt.
Faster!	Das Ausrufezeichen ist, wie viele Sonderzeichen, ungültig.
null, class	Der Name ist schon von Java belegt. Null – Groß-/Kleinschreibung ist relevant – oder cláss wären möglich.

Kapitel 2: Grundlagen der Programmierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Operatoren für Wahrheitswerte und numerische Datentypen
- Kontrollstrukturen für alternative Anweisungen: „if“-Anweisung, „if-else“-Anweisung und „switch“-Anweisung
- Kontrollstrukturen für Wiederholungen: „while“-Schleife, „do-while“-Schleife und „for“-Schleife

Lernziele:

- Arithmetische Operatoren, Vergleichsoperatoren und Boolsche Operatoren kennen und anwenden können
- Die Kontrollstrukturen „if“-Anweisung, „if-else“-Anweisung und „switch“-Anweisung kennen und anwenden können
- Die Kontrollstrukturen „while“-Schleife, „do-while“-Schleife und „for“-Schleife kennen und anwenden können

Operatoren

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Operatoren** sind Symbole, die verschiedene Operationen auf ihren Argumenten, den **Operanden**, ausführen.
- Eine mathematische Formel, etwa der Ausdruck –27 * 9, besteht aus **Operanden** und **Operatoren**. Operanden können auch Variablen sein.
- In Java dient das Gleichheitszeichen „= “ der Zuweisung einer Variable. Zuweisungen „sehen zwar so aus“ wie mathematische Gleichungen, doch gibt es einen wichtigen Unterschied:
  Die Formel a = a + 1 ist zwar mathematisch nicht zu erfüllen, aus Programmiersicht wird hier jedoch die Variable a um 1 erhöht.
- Wichtige Typen von Operatoren
  - Arithmetische Operatoren
  - Vergleichsoperatoren
  - Boolesche Operatoren

Arithmetische Operatoren (für numerische Datentypen)

TECHNISCHE UNIVERSITÄT DARMSTADT

Symbol	Name	Beispiel	Ergebnis
+	Addition	`int n = 1 + 2;`	`// n = 3`
-	Subtraktion	`int n = 42 - 23;`	`// n = 19`
*	Multiplikation	`int n = 2 * 4;`	`// n = 8`
/	Division	`int n = 43 / 6;`	`// n = 7`
%	Modulo	`int n = 42 % 10;`	`// n = 2`

Rangfolge: Für die Auswertung des Compilers gilt „Punkt vor Strich“ – zur besseren Übersicht sollten komplexe Ausdrücke geklammert werden.

Hinweis:

```
i++; // Abkürzung für i = i + 1
j--; // Abkürzung für j = j - 1
```

Beispiel: Datentypen und Operatoren

TECHNISCHE UNIVERSITÄT DARMSTADT

Was ist die Ausgabe des Programms?

```
public static void main(String[] args) {
	int ganzeZahl = 10 / 3;
	double kommaZahl = 10.0 / 1.5;

	System.out.println (ganzeZahl);
	System.out.println (kommaZahl);
}
```

Aufgabe: Anwendung von Operatoren

http://pingo.upb.de
#9456

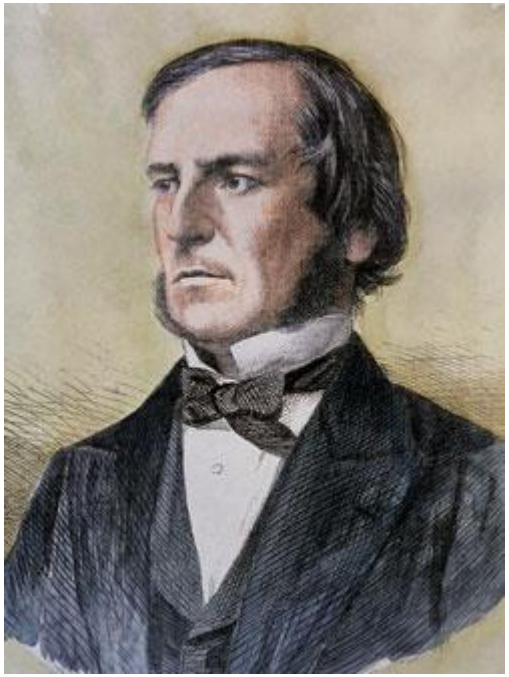
Welche Werte haben die Variablen a und b nach den Zuweisungen?

```
public static void main(String[] args) {
        int a = 10 * 4 - 6;
        int b = 10 + 100 % 4 - 6;
}
```

Wahrheitswerte: Der Datentyp Boolean

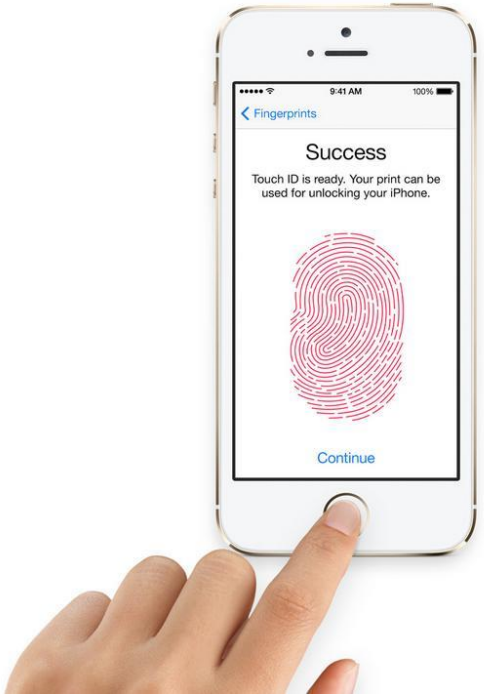
TECHNISCHE UNIVERSITÄT DARMSTADT

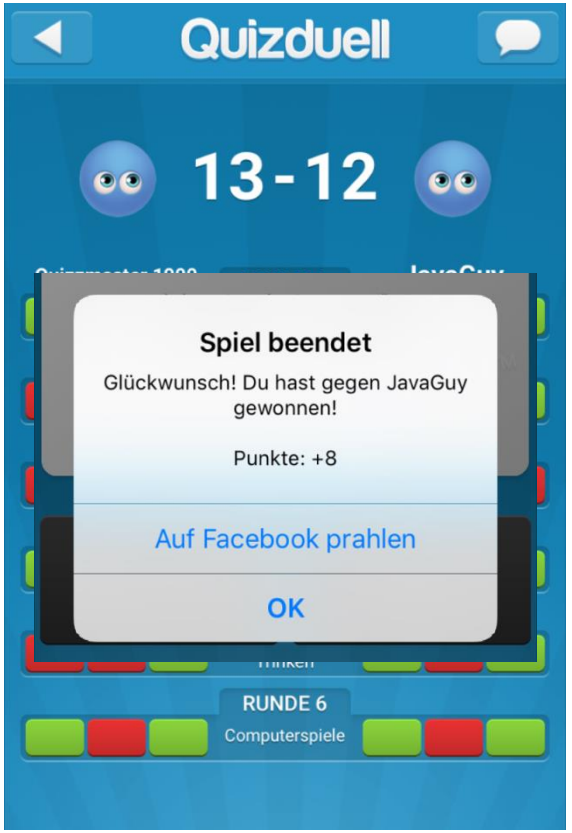
- Der **Datentyp boolean** beschreibt einen Wahrheitswert, der entweder true oder false ist.
- Der boolesche Typ wird beispielsweise bei Verzweigungen oder Schleifen benötigt.
- In der Regel ergibt sich ein Wahrheitswert aus **Vergleichen**.



George Boole

Vergleiche (Beispiele)





Vergleichsoperatoren

TECHNISCHE UNIVERSITÄT DARMSTADT

Das Ergebnis aller Vergleichsoperatoren ist ein Wert des Datentyps `boolean` (also `true` oder `false`)

Symbol	Bedeutung	Anwendung	Erklärung
==	gleich	`boolean b = (1 == 2);`	`// b = false`
!=	ungleich	`boolean b = (1 != 2);`	`// b = true`
>	größer	`boolean b = (1 > 2);`	`// b = false`
>=	größer gleich	`boolean b = (1 >= 2);`	`// b = false`
<	kleiner	`boolean b = (1 < 2);`	`// b = true`
<=	kleiner gleich	`boolean b = (1 <= 2);`	`// b = true`

Boolesche Operatoren

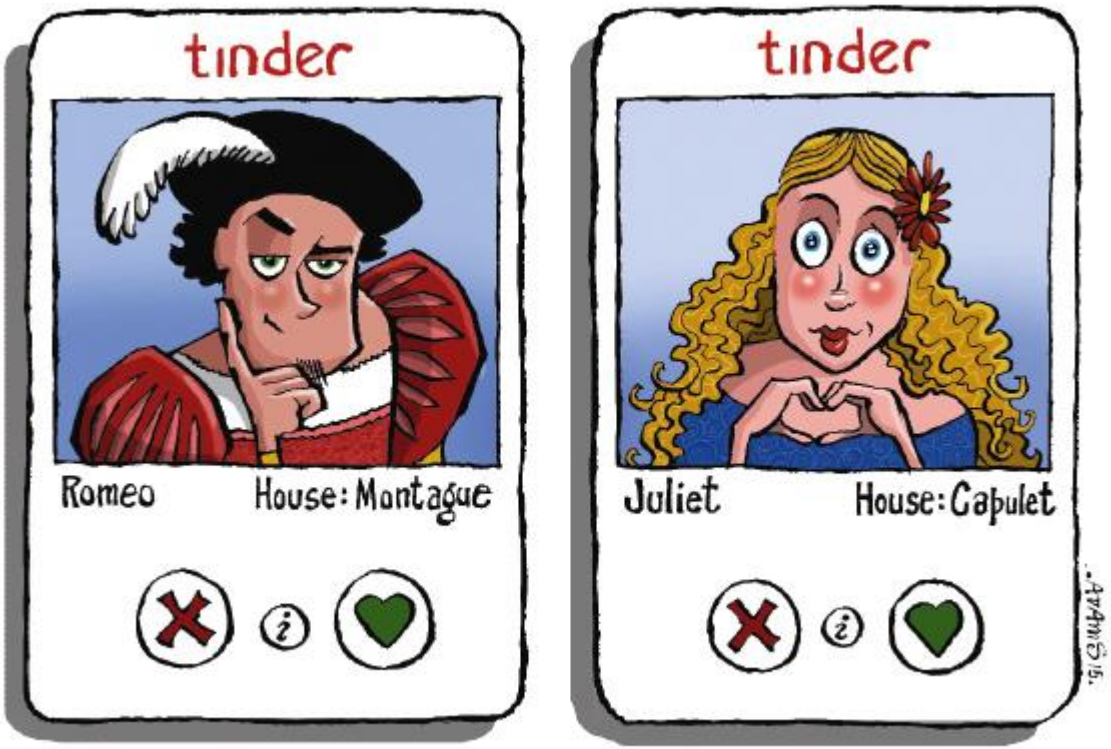
TECHNISCHE UNIVERSITÄT DARMSTADT

- Die Abarbeitung von Programmcode ist oft an Bedingungen geknüpft. Diese Bedingungen sind oftmals zusammengesetzt, wobei drei Operatoren verwendet werden:

  - **Nicht** (Negation): Aus wahr wird falsch und aus falsch wird wahr.
  - **Und** (Konjunktion): Beide Aussagen müssen wahr sein, damit die Gesamtaussage wahr wird.
  - **Oder** (Disjunktion): Eine der beiden Aussagen muss wahr sein, damit die Gesamtaussage wahr wird.

- Boolesche Operatoren operieren auf den Werten `true` und `false` und das Ergebnis ist wiederum ein Boolean

Boolsche Operatoren (Beispiel)



Details zu booleschen Operatoren

TECHNISCHE UNIVERSITÄT DARMSTADT

		Nicht	UND	ODER
Boolean a	**Boolean b**	! a	a && b	a || b
true	true	false	true	true
true	false	false	false	true
false	true	true	false	true
false	false	true	false	false

- Hinweise
  - `!` bindet stärker als `&&`
  - `&&` bindet stärker als `||`
- Beispiel:

```
boolean b = false || ! false && true // b = true
```

Symbol	Rang
!	1
&&	2
||	3

WIRTSCHAFTS INFORMATIK

Beispiele: Boolesche Ausdrücke

TECHNISCHE UNIVERSITÄT DARMSTADT

	(x > 3)	&&	(x <= 5)	
für x = 1	false	&&	true	= false
für x = 4	true	&&	true	= true

x = 5, b = true					
(x >= 3)	||	(x < 7)	&&	!b	
true		true	&&	false	
true	||	false			= true

x = 5, b = true							
(	(x >= 3)	||	(x < 7)	)	&&	!b	
	true	||	true			false	
	true				&&	false	= false

WIRTSCHAFTS INFORMATIK

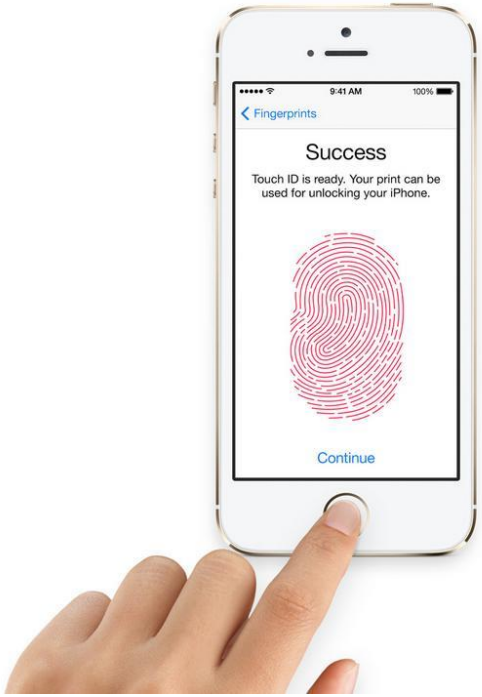
Kontrollstrukturen

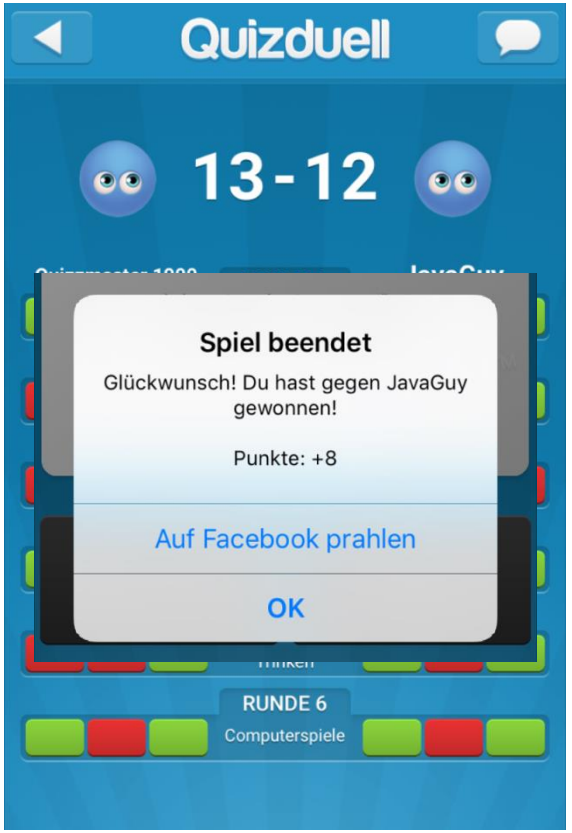
TECHNISCHE UNIVERSITÄT DARMSTADT

- Kontrollstrukturen dienen dazu, Programmteile unter bestimmten Bedingungen auszuführen.

- Zum Ausführen alternativer Programmteile (sog. **Verzweigungen**):
  - **if**-Anweisung
  - **if-else**-Anweisung
  - **switch**-Anweisung
- Neben der Verzweigung dienen **Schleifen** dazu, Programmteile mehrmals auszuführen:
  - **while**-Schleife
  - **do-while**-Schleife
  - **for**-Schleife

Vergleiche (Beispiele)





Alternativen: if-Anweisung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Eine „if“-Anweisung führt eine Anweisung aus, wenn die Bedingung (d. h. ein Boolean-Ausdruck) erfüllt (= wahr) ist.
- **Syntax**:
  - `if ( Bedingung ) {Anweisung}`
  - `if ( Bedingung ) {Anweisung1} else {Anweisung2}`
- Beispiel:

```
if ( x < 4 ) {
  System.out.println("x ist kleiner 4");
} else {
  System.out.println("x ist NICHT kleiner 4");
}
```

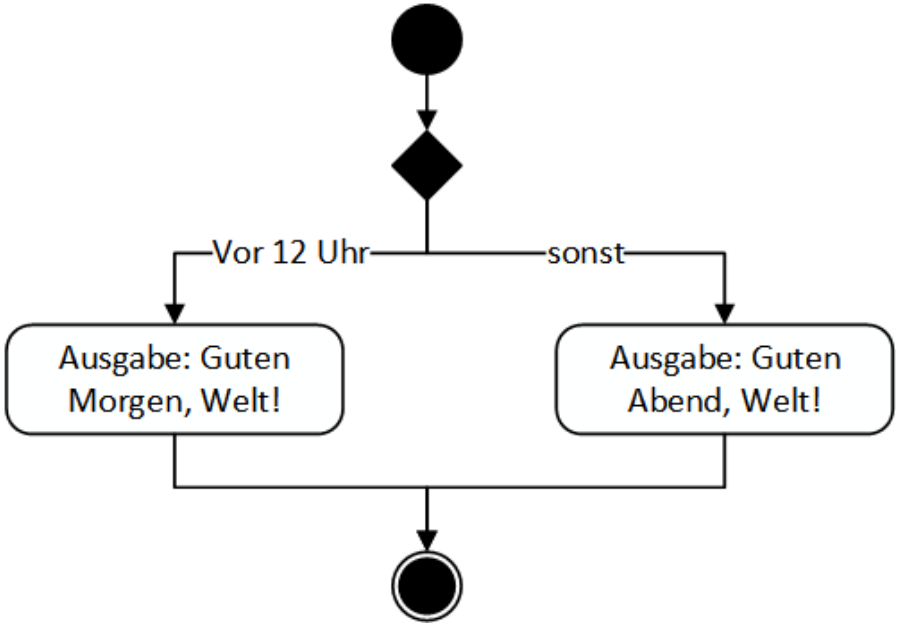
- **Hinweis:** Die Anweisung kann auch eine Blockanweisung oder eine weitere if-Anweisung („else if“) sein!

Beispiel: if-Anweisung

```
public class Alternative {
public static void
main(String[] args){
   int stunde=23;

   if (stunde < 12) {
   System.out.println(
   "Guten Morgen, Welt!");

   } else {
   System.out.println(
   "Guten Abend, Welt!");
   }
}
}
```



if-Anweisung: Bedingungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Syntax**:
  - `if ( Bedingung ) {Anweisung}`
  - `if ( Bedingung ) {Anweisung1} else {Anweisung2}`

- Die Bedingung einer if-Anweisung muss einem Boolean-Ausdruck entsprechen und einen Wahrheitswert (true oder false) als Ergebnis haben.
- Dabei kann der Boolean-Ausdruck auch aus mehreren verknüpften Ausdrücken und Befehlen zusammengesetzt sein.
- Beispiele:
  - `if (true) {Anweisung}`
  - `if (x < 4) {Anweisung}`
  - `if (false || ! false && true) {Anweisung}`
  - `if (x < 4 || ! false && true) {Anweisung}`

Vergleichsoperatoren

Das Ergebnis aller Vergleichsoperatoren ist ein Wert des Datentyps `boolean` (also `true` oder `false`)

Symbol	Bedeutung	Anwendung	Erklärung
==	gleich	`boolean b = (1 == 2);`	`// b = false`
!=	ungleich	`boolean b = (1 != 2);`	`// b = true`
>	größer	`boolean b = (1 > 2);`	`// b = false`
>=	größer gleich	`boolean b = (1 >= 2);`	`// b = false`
<	kleiner	`boolean b = (1 < 2);`	`// b = true`
<=	kleiner gleich	`boolean b = (1 <= 2);`	`// b = true`

Beispiele: Boolesche Ausdrücke

	(x > 3)	&&	(x <= 5)	
für x = 1	false	&&	true	= false
für x = 4	true	&&	true	= true

x = 5, b = true					
(x >= 3)	||	(x < 7)	&&	!b	
true		true	&&	false	
true	||	false			= true

x = 5, b = true							
(	(x >= 3)	||	(x < 7)	)	&&	!b	
	true	||	true			false	
	true				&&	false	= false

Aufgabe: if-Anweisungen

http://pingo.upb.de
#9456

Wie lautet die Bildschirmausgabe des Programms?

```
public static void main(String[] args){
boolean fehlerhaft = false;
boolean einwandfrei = true;
if ((!fehlerhaft && true) && (einwandfrei)) {
  System.out.println("hier ist ein Fehler");
} else {
  System.out.println("alles kein Problem");
}
}
```

Gruppieren von Anweisungen mit Blöcken

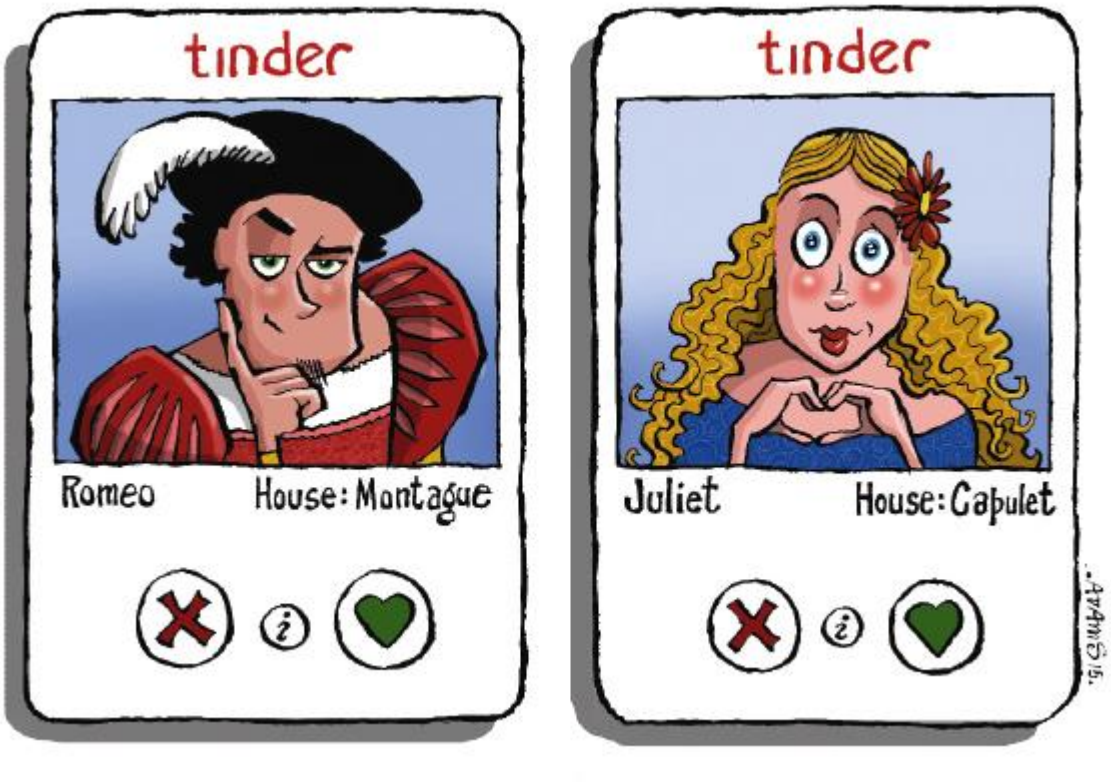
TECHNISCHE UNIVERSITÄT DARMSTADT

Ein **Block** fasst eine Gruppe von Anweisungen zusammen, die hintereinander ausgeführt werden. Anders gesagt: Ein Block ist eine Anweisung, die in geschweiften Klammern { } eine Folge von Anweisungen zu einer neuen Anweisung zusammenfasst:

```
{
Anweisung1;
Anweisung2;
...
}
```

Hinweis: Damit ist das Verschachteln von Kontrollstrukturen (und Blockanweisungen) möglich!

Boolsche Operatoren (Beispiel)



Alternativen: switch-Anweisung

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Switch-Anweisungen** ermöglichen eine kompakte Schreibweise
- **Syntax**:

```
switch (Ausdruck) {
  case Wert : Anweisung;
              Anweisung;
              break;
  case Wert 2: Anweisung;
  default : Anweisung;
}
```

- `Ausdruck` ist vom Typ `int` (oder `char`)
- Das Programm springt zu der **Marke**, die mit dem berechneten Wert übereinstimmt
- Stimmt keine Marke mit dem Wert überein, springt das Programm zur **default-Marke**

Bedeutung des break-Statements

TECHNISCHE UNIVERSITÄT DARMSTADT

- Der Befehl `break` sorgt dafür, dass direkt an das **Ende** der switch-Anweisung gesprungen wird
- Steht am Ende eines case-Blocks keine break-Anweisung, werden die Befehle in den nachfolgenden case-Blöcken ebenfalls ausgeführt, bis die nächste break-Anweisung erreicht wird
- Beispiel:

```
int x = 2;
switch (x) {
     case 1 : System.out.println("Eins"); break;
     case 2 :
     case 3 : System.out.println("Zwei oder Drei"); break;
     default : System.out.println("Unbekannt");
}
```

Beispiel: switch-Anweisung (Code)

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public class Sample2 {
  public static void main(String[] args) {
    int note = 1;
    String praed = "";
    switch (note) {
      case 1:
        praed = "sehr gut";
        break;
      case 2:
        ...
      case 6:
        praed = "ungenügend";
        break;
      default:
        praed = "Die Eingabe ist keine Note!";
        break;
    }
    System.out.println("Das Prädikat für die Note " +
      note + " lautet " + praed + ".");
  }
}
```