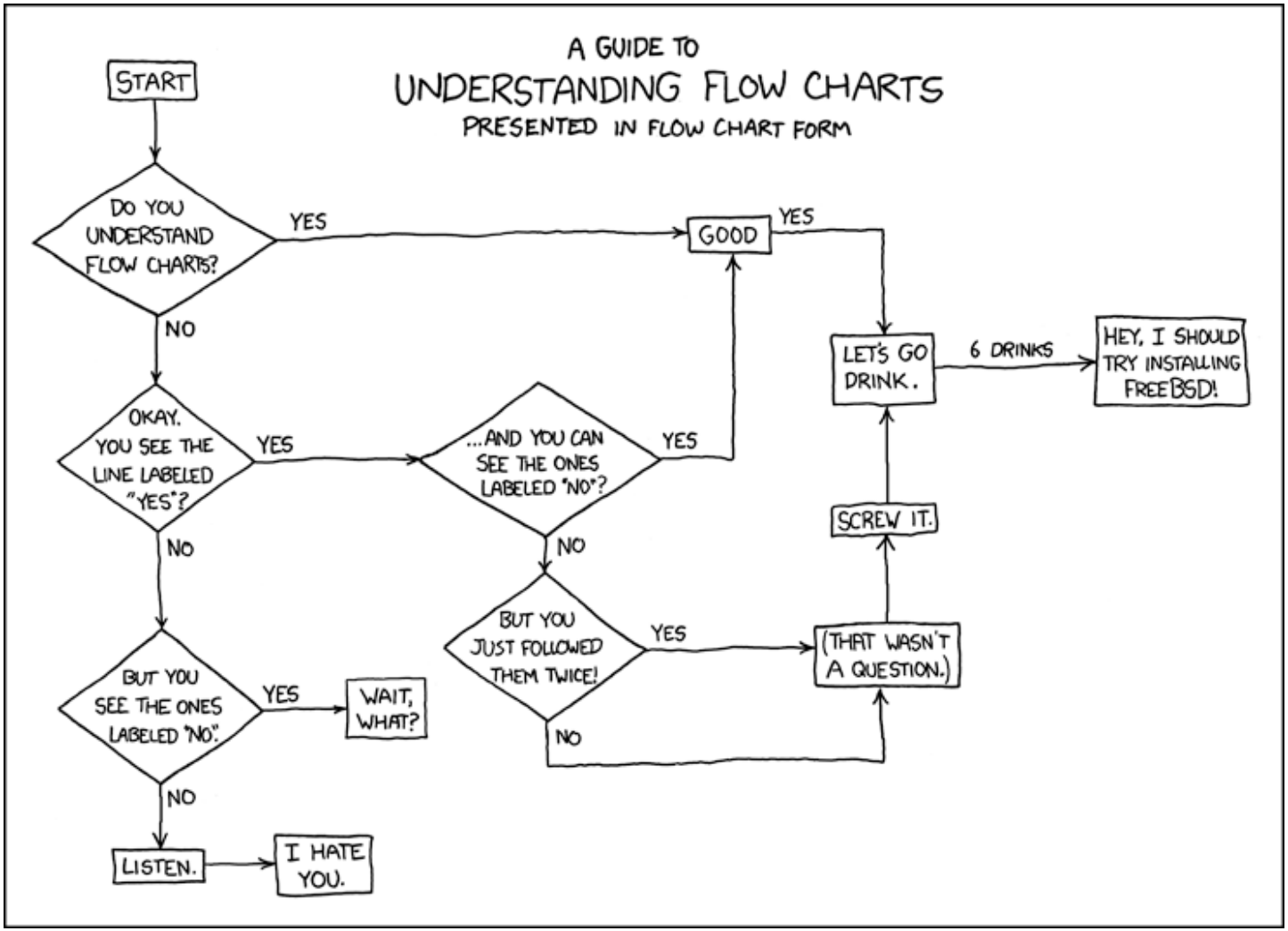
Universität Stuttgart
MINT-Kolleg Baden-Württemberg

Fallunterscheidungen

Vorkurs Informatik

Universität Stuttgart
Karlsruher Institut für Technologie
F. Schweiner
MINT-KOLLEG BADEN-WÜRTTEMBERG

Getting started ...



©Randall Munroe, `http://xkcd.com/518/`

Outline

(nach Folien von L. Vettin, V. Weidler, W. Kessler)

Outline

Aussagen

Aussage

Eine Aussage ist ein Satz, der entweder wahr (true; 1) oder falsch (false; 0) ist.

Aussagen

Aussage

Eine Aussage ist ein Satz, der entweder wahr (true; 1) oder falsch (false; 0) ist.

Beispiele:

- Heute ist Montag.
- Rosen sind Tiere.
- 42 ist größer als 17.
- Alle Primzahlen sind ungerade.

Aussagen

Aussage

Eine Aussage ist ein Satz, der entweder wahr (true; 1) oder falsch (false; 0) ist.

Beispiele:

- Heute ist Montag.
- Rosen sind Tiere.
- 42 ist größer als 17.
- Alle Primzahlen sind ungerade.

Achtung: Nicht alles, was gesagt werden kann, ist (mathematisch gesehen) eine Aussage!

Logische Operatoren

Einzelne Aussagen lassen sich durch logische Operatoren verknüpfen:

UND Die Konjunktion zweier Aussagen ist dann wahr, wenn *beide* Aussagen wahr sind. Andernfalls ist sie falsch.
Schreibweise Konjunktion von a und b: $a \wedge b$

ODER Die Disjunktion zweier Aussagen ist dann wahr, wenn *mindestens eine* der beiden Aussagen wahr ist. Sie ist falsch, wenn beide Aussagen falsch sind.
Schreibweise Disjunktionvon a und b: $a \vee b$

NICHT Die Negation einer Aussage ist dann wahr, wenn die Aussage falsch war, und umgekehrt.
Schreibweise Negation von a: $\neg a$

Alltagssprache und Logik

Die Bedeutung der logischen Operatoren und die Verwendung von “und” / “oder” in der Alltagssprache stimmen nicht immer überein:

- a: ‘Ich gehe ins Schwimmbad” und b:“Ich gehe ins Kino”.
  Wann ist $a \wedge b$ wahr?

  Alltag Auch wenn erst a und danach b wahr ist.
  Logik Nur wenn a und b *gleichzeitig* wahr sind.

Alltagssprache und Logik

Die Bedeutung der logischen Operatoren und die Verwendung von "und" / "oder" in der Alltagssprache stimmen nicht immer überein:

- a: 'Ich gehe ins Schwimmbad" und b:"Ich gehe ins Kino".
  Wann ist $a \wedge b$ wahr?

  Alltag Auch wenn erst a und danach b wahr ist.
  Logik Nur wenn a und b *gleichzeitig* wahr sind.

- a: 'Ich bin ein Mensch" und b:"Ich bin ein Vogel".
  Wann ist $a \vee b$ wahr?

  Alltag Wenn *entweder* a oder b wahr ist,
  aber meist kann nicht beides wahr sein.
  Logik Wenn a, b oder beides wahr ist.

Quiz: Aussagenlogik

Gegeben seien folgende Wahrheitswerte für die Aussagen a und b:

a wahr bzw. 1
b falsch bzw. 0

Entscheiden Sie: Welche Aussagen sind wahr?

- $a \wedge b$
- $a \vee b$
- $\neg b$
- $b \wedge \neg b$
- $a \vee \neg a$
- $b \wedge \neg a$

Quiz: Aussagenlogik

Gegeben seien folgende Wahrheitswerte für die Aussagen a und b:

a	wahr bzw. 1
b	falsch bzw. 0

Entscheiden Sie: Welche Aussagen sind wahr?

- $a \wedge b$ (falsch)
- $a \vee b$ (wahr)
- $\neg b$ (wahr)
- $b \wedge \neg b$ (immer falsch)
- $a \vee \neg a$ (immer wahr)
- $b \wedge \neg a$ (falsch)

Aussagen (Wahrheitswerte) in Java

- Wahrheitswerte haben in Java den Datentyp `boolean`.
- Sie können nur die zwei Werte `true` und `false` annehmen.

Aussagen (Wahrheitswerte) in Java

- Wahrheitswerte haben in Java den Datentyp `boolean`.
- Sie können nur die zwei Werte `true` und `false` annehmen.
- Wahrheitswerte können durch Variablenvergleiche entstehen:

Symbol	Bedeutung
==	gleich
>	größer
<	kleiner

Symbol	Bedeutung
!=	ungleich
>=	größer gleich
<=	kleiner gleich

- Beispiele:

```
boolean doorIsOpen = true;
boolean numberIsNegative = (number < 0);
boolean numberIsPositiveAndOdd = (number % 2 == 1);
boolean valuesAreNotEqual = (number1 != number2);
```

Logische Operatoren in Java

- Java kennt drei logische Operatoren:

Symbol	Bedeutung
&&	$\wedge$ UND
||	$\vee$ ODER
!	$\neg$ NICHT

- Es gibt eine Priorisierung: `!` vor `\&\&` vor `||`.
- Klammern legen auch die Reihenfolge fest.
- Beispiele:

```
boolean isNotNegative = !(number < 0);
boolean isOddAndNegative =
      (number % 2 == -1) && (number < 0);
boolean isEvenOrNotNegative =
      (number % 2 == 0) || !(number < 0);
```

Quiz: Wahrheitswerte

Bei welchen Zahlen wird dieser Ausdruck wahr:

```
(number % 3 == 0) && (number > 0 || number == -16)
```

- -16, 3, 6, 9, 12, 15, ...
- 3, 6, 9, 12, 15, ...
- -16, 1, 2, 4, 5, 7, 8, 10, 11, ...
- 1, 2, 4, 5, 7, 8, 10, 11, ...
- -3, -6, -9, -12, -15, -16, ...
- -3, -6, -9, -12, -15, -18 ...

Quiz: Wahrheitswerte

Bei welchen Zahlen wird dieser Ausdruck wahr:

```
(number % 3 == 0) && (number > 0 || number == -16)
```

- -16, 3, 6, 9, 12, 15, ...
- 3, 6, 9, 12, 15, ...
- -16, 1, 2, 4, 5, 7, 8, 10, 11, ...
- 1, 2, 4, 5, 7, 8, 10, 11, ...
- -3, -6, -9, -12, -15, -16, ...
- -3, -6, -9, -12, -15, -18 ...

Lösung: 3, 6, 9, 12, 15, ...

Outline

`if`-Anweisung

`if`-Anweisung

Die `if`-Anweisung verwendet man für Programmteile, die nur ausgeführt werden sollen, wenn eine bestimmte Bedingung erfüllt ist.

```
int age = 20;
if ( age < 18 ) {
   System.out.println("Your are underage.");
}
```

- Nach dem Schlüsselwort `if` steht in Klammern eine Bedingung.
- Die Bedingung ist vom Typ Wahrheitswert (`boolean`).
- Die Anweisungen in {} Klammern werden nur ausgeführt, wenn die Bedingung zutrifft.

`if`-Anweisung

`if`-Anweisung

Die `if`-Anweisung verwendet man für Programmteile, die nur ausgeführt werden sollen, wenn eine bestimmte Bedingung erfüllt ist.

```
int number = 4;
boolean numberIsNegative = ( number < 0 );

if ( numberIsNegative ) {
   System.out.println(" ... ");
}
```

- In den runden Klammern kann auch eine `boolean`-Variable stehen.
- Verwendet man aussagenkräftige Namen, kann man die Anweisung wie einen Text lesen: „If number is negative then ...“.

Beispiele mit `if`

```
if ( age >= 18 && age <= 30) {
   System.out.println("Your age is between 18 and 30.");
   System.out.println("It is: " + age);
}
```

Beispiele mit `if`

```
if ( age >= 18 && age <= 30) {
   System.out.println("Your age is between 18 and 30.");
   System.out.println("It is: " + age);
}
```

```
if ( true ) {
   System.out.println("Printed in any case.");
}
```

Beispiele mit `if`

```
if ( age >= 18 && age <= 30) {
   System.out.println("Your age is between 18 and 30.");
   System.out.println("It is: " + age);
}
```

```
if ( true ) {
   System.out.println("Printed in any case.");
}
```

```
int personsInPool = 0;
boolean iGoSwimming = ...;

if ( iGoSwimming ) {
   personsInPool += 1;
}
```

Häufige Fehlerquellen bei `if`

- Vergessen der {} Klammern:
  - Dies ist erlaubt. Dann wird aber nur die *eine* nächste Zeile nach dem `if` so betrachtet, als ob sie in {} steht.
  - Einrückungen spielen keine Rolle.

```
int age = 18;
if ( age > 21 ) // FALSCH: keine {}
   System.out.println("Is printed if age > 21.");
   System.out.println("Not part of the if.");
```

Häufige Fehlerquellen bei `if`

- Vergessen der {} Klammern:
  - Dies ist erlaubt. Dann wird aber nur die *eine* nächste Zeile nach dem `if` so betrachtet, als ob sie in {} steht.
  - Einrückungen spielen keine Rolle.

```
int age = 18;
if ( age > 21 ) // FALSCH: keine {}
   System.out.println("Is printed if age > 21.");
   System.out.println("Not part of the if.");
```

- Semikolon ; nach der Bedingung:
  - Semikolon ; beendet Statements, hier das *komplette* `if`.
  - Was folgt, wird nicht als Teil des `if`-Statements betrachtet.

```
int age = 18;
if ( age == 21 ); { // FALSCH: ;
   System.out.println("Not part of the if.");
}
```

Häufige Fehlerquellen bei Vergleichen

- Vergleiche vs. Zuweisungen:
  - In Java ist $=$ der Zuweisungsoperator,
  - Gleichheit wird mit $==$ geprüft.

```
int number = 1; // Zuweisung mit =
if ( number == 1 ) { // Vergleich mit ==
   System.out.println("number is one.");
}
```

Häufige Fehlerquellen bei Vergleichen

- Vergleiche vs. Zuweisungen:
  - In Java ist = der Zuweisungsoperator,
  - Gleichheit wird mit == geprüft.

```
int number = 1; // Zuweisung mit =
if ( number == 1 ) { // Vergleich mit ==
   System.out.println("number is one.");
}
```

- Gleichheit von Strings:
  - Nur elementare Datentypen mit == vergleichen.
  - Für Strings verwendet man `"text1".equals("text2")`.

```
String name = "Klaus";
String password = "1wx";

if ( name.equals("Bernd") && password.equals("1xw") ) {
   System.out.println("Login successful");
}
```

`if`-Anweisungen

```
int x = 42;
if (???) { System.out.println("Hello World"); }
```

Überlegen Sie sich, wie der Ausdruck in der `if`-Anweisungen aussehen muss, damit `Hallo Welt` nur dann ausgegeben wird, wenn

- `x` größer als Null ist
- `x` durch 3 teilbar ist
- `x` ungleich 32 ist
- `x` größer als 3 und kleiner als 10 ist
- `x` größer als 5 oder kleiner als 1 ist
- `x` größer null und durch 3 teilbar ist oder gleich -2 ist

if-Anweisungen

```
int x = 42;
if (???) { System.out.println("Hello World"); }
```

- `x>0`
- `x%3==0`
- `x!=32`
- `x>3 && x<10`
- `x>5 || x<1`
- `(x>0 && x%3==0)||(x==-2)`

Zweiseitige Fallunterscheidung

`if`/`else`-Anweisung

Die `if`/`else`-Anweisung führt ein Codefragment aus, wenn die angegebene Bedingung wahr ist, und ein anderes Codefragment, wenn die Bedingung falsch ist.

```
if (age >= 18) { // age is 18 or higher
   System.out.println("You are of legal age");
}
else { // age is lower than 18
   System.out.println("You are underage.");
}
```

Mehrfachverzweigung

Nach einem `else` kann wieder ein `if` stehen, das selbst wieder ein `else` hat und so weiter.

```
if ( age >= 21 ) {
   // older than 21
   System.out.println("You are really grown up.");
}
else if ( age > 17 ) {
   // between 18 and 21
   System.out.println("You are of legal age.");
}
else {
   // younger than 18
   System.out.println("You are underage.");
}
```

Syntax von `if/else`

Syntax

Die abstrakten Regeln, nach denen Ausdrücke in einer Programmiersprache gebildet werden

- Syntax gibt keine lauffähigen Codeblöcke sondern beschreibt die abstrakte Struktur einer Anweisung.
- Die Beschreibung enthält Platzhage die in <> eingeschlossen werden, z. B. `<Bedingung>`, `<Anweisung>`.
- Diese Teile müssen im Programm durch geeigneten Code ersetzt werden, z. B. `<Bedingung>` durch `age > 18` oder `<Anweisung>` durch `System.out.println("...");`.
- Optionale Teile sind durch `// (optional)` markiert.

Syntax von `if/else`

```
if ( <Bedingung> ) {
   // Diese Anweisungen werden ausgeführt wenn die
   // Bedingung zutrifft
   <Anweisung>;
   ...
}
else { // (optional)
   // Diese Anweisungen werden ausgeführt wenn die
   // Bedingung NICHT zutrifft
   <Anweisung>;
   ...
}
```

Outline

Zusammenfassung

- Eine **Aussage** ist ein Satz, der entweder wahr (true; 1) oder falsch (false; 0) ist.
- Variablenwerte von elementaren Datentypen können mit `==`, `>`, `<`, usw. **verglichen** werden, Objekttypen mit `.equals()`.
- **Wahrheitswerte** haben in Java den Datentyp `boolean`.
- Wahrheitswerte können mit den logischen **Operatoren** `&&` (UND), `||` (ODER) und `!` (NICHT) verknüpft werden.
- Die **if/else**-Anweisung führt ein Codefragment aus, wenn die angegebene Bedingung wahr ist, und ein anderes Codefragment, wenn die Bedingung falsch ist.

Universität Stuttgart
MINT-Kolleg Baden-Württemberg

Vielen Dank!

Frank Schweiner

E-Mail frank.schweiner@mint-kolleg.de
Telefon +49 (0) 711 685-84326
Fax —

Universität Stuttgart
MINT-Kolleg Baden-Württemberg

Azenbergstr. 12
70174 Stuttgart