

# INFORMATIK I

Tutorium 8 — 06. Dezember 2024



Name Last  
Universität Münster



L<sup>A</sup>T<sub>E</sub>X-Vorlage von  
Florian Sihler

Ich muss noch ein wenig schimpfen...



# TRUE-VERGLEICHE

Warum ist der folgende Code schlecht?

```
negate :: Bool -> Bool
negate b
  | (b == True)   = False
  | (b == False)  = True
```

Prüft Booleans

Diagram illustrating the code and its execution:

- The function signature `negate :: Bool -> Bool` is shown.
- The function definition `negate b` is shown.
- The function body consists of two guarded equations:
  - `| (b == True) = False`
  - `| (b == False) = True`
- Arrows indicate the evaluation of the guard expressions:
  - From `(b == True)` to `(True == True) = True = b`
  - From `(b == False)` to `(True == False) = False = not b`

- Wir vergleichen Wahrheitswerte und erhalten Wahrheitswerte.
  - › Boole'sche Operatoren

# BEDINGUNGEN UND ALTERNATIVEN

Wie können wir den Code verbessern?

```
negate :: Bool -> Bool
```

```
negate b
```

```
  | (b)      = False
```

```
  | (not b)  = True
```

 b ist nicht **True**, also bleibt uns nur noch ein Fall

- Wir vergleichen Wahrheitswerte und erhalten Wahrheitswerte.
  - Boole'sche Operatoren
- Wir können redundante Vergleiche reduzieren.
  - otherwise (oder else)

# (REDUNDANTE) BOOLEAN-RÜCKGABE

Warum ist das immernoch nicht perfekt?

```
negate :: Bool -> Bool
```

```
negate b
```

```
  | b           = False
```

```
  | otherwise = True
```

Wir erinnern uns:

```
Booleans == True = False
```

```
  | (not b) == True = True
```

- Wir vergleichen Wahrheitswerte und erhalten Wahrheitswerte.
  - Boole'sche Operatoren
- Wir können redundante Vergleiche reduzieren.
  - otherwise (oder else)
- Wir geben Wahrheitswerte explizit zurück.
  - Boole'sche Operatoren

## Schlechter Code

```
negate :: Bool -> Bool
negate b
  | (b == True)   = False
  | (b == False) = True
```

## Verbesserter Code

```
negate :: Bool -> Bool
negate b = not b
```

Aufgabe: Vereinfachen Sie den folgenden Java-Code.

```
public boolean nand(boolean a, boolean b) {  
    if (a == false) return true;  
    else if (a == true) {  
        if (b == false) return true;  
        if (b == true) return false;  
    }  
}
```

# ÜBUNGSAUFGABE

Lösung: `public boolean nand(boolean a, boolean b) {...}`

```
if (a == false) return true;
else if (a == true) {
    if (b == false) return true;
    if (b == true) return false;
}

if (!a) return true;
else if (a) {
    if (!b) return true;
    if (b) return false;
}
```




# ÜBUNGSAUFGABE

Lösung: `public boolean nand(boolean a, boolean b) {...}`

```
if (a == false) return true;
else if (a == true) {
    if (b == false) return true;
    if (b == true) return false;
}
```

```
if (!a) return !a;
else if (a) {
    if (!b) return (!b);
    if (b) return (!b);
}
```

redundant




# ÜBUNGSAUFGABE

Lösung: `public boolean nand(boolean a, boolean b) {...}`

```
if (a == false) return true;
else if (a == true) {
    if (b == false) return true;
    if (b == true) return false;
}
```

Das können wir zusammenfassen!

```
if (!a) return !a;
else if (a) {
    return (!b);
}
```



# ÜBUNGSAUFGABE

Lösung: `public boolean nand(boolean a, boolean b) {...}`

```
if (a == false) return true;
else if (a == true) {
    if (b == false) return true;
    if (b == true) return false;
}
```

```
if (a) {
    return (!b);
} else
return !a;
```

} Kombination durch AND

# ÜBUNGSAUFGABE

Lösung: `public boolean nand(boolean a, boolean b) {...}`

```
if (a == false) return true;
else if (a == true) {           // if (a)
    if (b == false) return true;   return (a && !b);
    if (b == true) return false;  }
}                                // else
                                return !a;
```

Kombination  
durch OR

# ÜBUNGSAUFGABE

Lösung: `public boolean nand(boolean a, boolean b) {...}`

```
if (a == false) return true;
else if (a == true) {
    if (b == false) return true;
    if (b == true) return false;
}
```

`return (a && !b) || !a;`

Das geht einfacher!

# ÜBUNGSAUFGABE

Lösung: `public boolean nand(boolean a, boolean b) {...}`

```
if (a == false) return true;
else if (a == true) {
    if (b == false) return true;
    if (b == true) return false;
}
```

`return !(a && b);`

Vergleich: Blatt 5 Aufgabe 2 a)

```
nand :: Bool -> Bool -> Bool
nand a b = not (a && b)
```

# Java ist auch eine Insel!



## Aufgabe 1

Erstellen Sie einen Konfigurationsdialog für einen Würfel, bei dem der Benutzer die gewünschte Seitenlänge (in Zentimetern) und das gewünschte Material angibt, und berechnen Sie das Gewicht (in Kilogramm) und die Haftreibung des Würfels mit drei Nachkommastellen.

Verfügbare Materialien und deren Dichte:

- Plastik:  $5 \text{ kg/m}^3$
- Holz:  $15 \text{ kg/m}^3$
- Eisen:  $25 \text{ kg/m}^3$

Für andere Eingaben wird Plutonium mit einer Dichte von  $10000 \text{ kg/m}^3$  verwendet.



## Lösung 1

```
public static void printIntroduction() {  
    System.out.println("Hello, _test_subject._Welcome_to_Aperture_Science's_computer-aided↵  
        _enrichment_center.");  
    System.out.println("Before_we_begin,_we_must_configure_your_Aperture_Science_Weighted↵  
        _Storage_Cube.");  
}  
  
public static int getCubeLength() {  
    System.out.print("\nPlease_input_your_desired_length_(in_centimeters):_");  
    return IOTools.nextInt();  
}  
  
public static double calculateVolume(int lengthCm) {  
    double lengthMeters = lengthCm / 100.0;  
    return lengthMeters * lengthMeters * lengthMeters;  
}
```

## Lösung 1

```
public static int getMaterialDensity() {  
    int material = IOTools.nextInt();  
  
    int density;  
    switch (material) {  
        case 1:  
            density = DENSITY_PLASTIC; break;  
        case 2:  
            density = DENSITY_WOOD; break;  
        case 3:  
            density = DENSITY_IRON; break;  
        default:  
            density = DENSITY_PLUTONIUM; break;  
    }  
  
    return density;  
}
```

## Lösung 1

```
public static double calculateWeight(int density, double volume) {  
    return density * volume;  
}  
  
public static double calculateStiction(double weight) {  
    double gravity = 9.81;  
    double stictionCoefficient = 1.0;  
    double stiction = weight * gravity * stictionCoefficient;  
    return stiction;  
}  
  
public static double roundToThreeDecimals(double value) {  
    return (double) ((int) (value * 1000)) / 1000;  
}
```

# WELL DONE, ANDROID.

## Lösung 1

Hello, test subject. Welcome to Aperture Science's computer-aided enrichment center.  
Before we begin, we must configure your Aperture Science Weighted Storage Cube.

Please input your desired length (in centimeters): 13

Excellent. Now, please choose a material for your cube.

- 1 - Plastic
- 2 - Wood
- 3 - Iron

Awaiting selection...: 7

The Enrichment Center regrets to inform you that your selection is... impossible.  
As a result, we have chosen Lead-coated Plutonium for you. It's terribly unsafe, but you're a test subject,  
not a survivor.

```
-----  
Cube Dimensions: 13x13x13 cm^3  
Cube Weight:      21.97 kg  
Cube Stiction:    215.525 kg  
-----
```

Please proceed to the chamberlock. And don't worry... you'll be fine... probably.

# NO LEMONS, NO MELON.

## Aufgabe 2

Erstellen Sie ein Programm, das einen String einliest und prüft, ob es sich um ein Palindrom handelt.

## Lösung 2

```
public static boolean isPalindrome(String palindrome) {  
    char[] letters = palindrome.toLowerCase().toCharArray();  
  
    for (int i = 0; i < letters.length / 2; i++) {  
        if (letters[i] != letters[letters.length - 1 - i]) {  
            return false; // return on first mismatch  
        }  
    }  
    return true;  
}
```

## Aufgabe 3

Erstellen Sie ein Java-Programm, das Hexadezimal-Strings in Dezimalzahlen umwandelt.

## Lösung 3: Ausgabe aller gültigen Zeichen

```
char[] validChars = ↵  
    {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};  
  
System.out.print("Gültige Zeichen: ");  
for (char c : validChars) {  
    System.out.print(c + ", ");  
}  
System.out.println();
```

- ▶ Die magische Zahl CAFEBABE am Anfang von Java-Bytecode-Dateien kennzeichnet sie für die Ausführung durch die Java-VM.
- ▶ In der Virtualisierungssoftware XEN war B00B135 früher eine Benutzer-ID, die für den Zugriff auf das System erforderlich war.

## Lösung 3: Gültigkeit prüfen

```
if (!isValidHexArray(hexArray, validChars))
    System.out.println("Der String " + input + " ist kein gültiger Hex-String.");

public static boolean isValidHexArray(char[] hexArray, char[] validChars) {
    for (char c : hexArray) {
        if (!isValidHexChar(c, validChars)) return false;
    }
    return true;
}

public static boolean isValidHexChar(char c, char[] validChars) {
    for (char validChar : validChars) {
        if (c == validChar) return true;
    }
    return false;
}
```

## Lösung 3: Dezimalzahl berechnen

```

long decimalValue = hexToDecimal(hexArray, validChars);

public static long hexToDecimal(char[] hexArray, char[] validChars) {
    long result = 0;
    int length = hexArray.length;
    for (int i = 0; i < length; i++) {
        int hexValue = findHexValue(hexArray[i], validChars);
        result += hexValue * Math.pow(16, length - 1 - i);
    }
    return result;
}

public static int findHexValue(char hexChar, char[] validChars) {
    for (int i = 0; i < validChars.length; i++) {
        if (hexChar == validChars[i]) return i;
    }
    return -1;
}

```



## Lösung 3: Alternative

```
long decimalValue = hexToDecimalWithChars(hexArray);

public static long hexToDecimalWithChars(char[] hexArray) {
    long result = 0;
    for (char hexChar : hexArray) {
        int hexValue = getHexValue(hexChar);
        result = result * 16 + hexValue;
    }
    return result;
}

public static int getHexValue(char hexChar) {
    if (Character.isDigit(hexChar)) {
        return hexChar - '0';
    } else {
        return hexChar - 'a' + 10;
    }
}
```

## Aufgabe 4

Implementieren Sie (schon wieder) den Euklidischen Algorithmus.

## Lösung 4

```
// Test auf negative Zahlen:  
if (m < 0 || n < 0) System.out.println("Nicht-negative_Zahlen_erwartet!");  
  
// Aus Pseudocode übernommener Algorithmus:  
private static int calculateGCD(int m, int n) {  
    if (m == 0) return n;  
    while (n != 0) {  
        if (m > n) m = m - n;  
        else n = n - m;  
    }  
    return m;  
}
```

# Von Päckchen und Paketen...

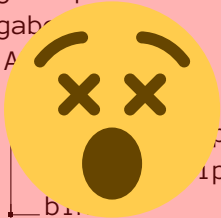


# ARCHIVE: ZIP

```
/
├── Abgabe.pdf
├── Abgabe.zip
│   ├── Code.java
│   └── Euclid.java
```

```
/
├── Abgabe.pdf
├── Abgabe.zip
│   ├── __MACOSX
│   │   ├── ._Code.java
│   │   └── ._Euclid.java
│   └── Abgabe
│       ├── Code.java
│       └── Euclid.java
```

```
/
├── Abgabe.pdf
├── Abgabe.zip
│   ├── A
│   │   ├── bin
│   │   │   ├── Code.class
│   │   │   └── Euclid.class
│   │   └── lib
│   │       └── Prog1Tools.jar
│   └── src
│       └── Code.java
```



# ARCHIVE: ZIP

## Kommandozeile (CLI)

```
zip archive.zip file0 file1 file2
```

```
# directories
```

```
zip -r archive.zip directory0 directory1 # file0 file1 ...
```

## Visuell (GUI)

- 7zip
- Explorer/Finder/FileManager/...
- ...

Live on, Time. Emit no evil.

**Name Last**

Münster, 5. Dezember 2024

name.last@uni-muenster.de