Recap

- Functions: Named code blocks
  - Can be called repeatedly
  - Can have arguments that change their behaviour
    - Arguments are accessible as variables within the body of the function
  - Functions can return a value
    - Type needs to be declared

return

Section 1

Exercise 3

Exercises

- Reference solutions available in Ilias
- No individual feedback
  - You should compare your solution with the reference solution and see if you understand everything
  - Ask in the tutorial if you have questions
- New stuff every week: Difficult to catch on if you lost your footing
- ⚠ Exercises are meant to take some time

Exercises

- Reference solutions available in Ilias
- No individual feedback
  - You should compare your solution with the reference solution and see if you understand everything
  - Ask in the tutorial if you have questions
- New stuff every week: Difficult to catch on if you lost your footing
- ⚠ Exercises are meant to take some time

How to start

- Add a (empty) main function
  - `public static void main(String[] args)`
  - Verify that no error messages/warnings are shown
- If you have one error message: Fix it before continuing

Exercise 03: func1

```
1  public class Exercise03 {
2    public static void main(String[] args) {
3      System.out.println(func1(3.5, 2.0)); // 1
4      System.out.println(func1(12, 5.3)); // 9
5      System.out.println(func1(35, 20)); // 100
6    }
7
8    static int func1(double d1, double d2) {
9      return (int) (d1 * d2 / 7f);
10   }
11 }
```

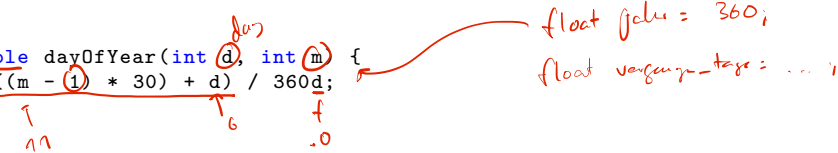
double int
d1 * d2 / 7

double d3 = d1 * d2;
double d4 = d3 / 7d;
int i = (int) d4;
return i;

Exercise 03: dayOfYear

```java
public class Exercise03 {
  public static void main(String[] args) {
    System.out.println(dayOfYear(1, 1)); // 0.002777777777777778
    System.out.println(dayOfYear(30, 12)); // 1.0
    System.out.println(dayOfYear(6, 11)); // 0.85
  }

  static double dayOfYear(int d, int m) {
    return (((m - 1) * 30) + d) / 360d;
  };
}
```



Exercise 03: isOdd

```java
public class Exercise03 {

  public static void main(String[] args) {
    System.out.println(isOdd(3)); // true
    System.out.println(isOdd(1)); // true
    System.out.println(isOdd(457483841)); // true
    System.out.println(isOdd(12)); // false
  }

  static boolean isOdd(int number) {
    return (number % 2) == 1;
  }
}
```

number = 17 17 % 2 = 1 == 1

number % 2 == 1;

Operator precedence

Noteworthy

- There can only be one main function (per file)
  - Recommendation: Always define a single main function, and call other functions from it
- A program can run perfectly fine without producing output
  - It's our job to let it produce output (e.g., by adding `System.out.println()`)
- Returning something and printing it on the console are two different things
  - Exercise 3: Functions should return something
  - In general: Good idea to separate internal logic from how it is shown to users
    - I.e., have a separate part of the program handle the output

```java
public class Exercise03 {

  public static void main(String[] args) {
    System.out.println(isOdd(12));

    boolean b = isOdd(13);
    System.out.println(!b);

    double doy = dayOfYear(6, 11);
    System.out.println(doy);
  }

  static boolean isOdd(int number) {
    return (number % 2) == 1;
  }

  static double dayOfYear(int d, int m) {
    int days = (((m - 1) * 30) + d)
    double returnValue = days / 360d;
    return returnValue;
  }
}
```

```java
public class Exercise03 {

  public static void main(String[] args) {
    System.out.println(isOdd(12));

    boolean b = isOdd(13);
    System.out.println(!b);

    double doy = dayOfYear(6, 11);
    System.out.println(doy);
  }

  static boolean isOdd(int number) {
    return (number % 2) == 1;
  }

  static double dayOfYear(int d, int m) {
    int days = (((m - 1) * 30) + d);
    double returnValue = days / 360d;
    return returnValue;
  }
}
```

Let's colorize all

- variable assignments
- variable declarations
- variable uses
- expressions
- statements
- function calls
- function declarations

UNIVERSITÄT
ZU KÖLN

Session 4: Conditionals

Softwaretechnologie: Java I

Nils Reiter
nils.reiter@uni-koeln.de

November 6, 2024

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

Section 2

Conditionals

Conditionals

- So far: All statements are executed in sequence
- Conditionals allow specifying a condition: If it is fulfilled, a statement is executed

Conditionals

- So far: All statements are executed in sequence
- Conditionals allow specifying a condition: If it is fulfilled, a statement is executed
- Two forms:

```
if (EXPRESSION) STATEMENT
```

```
if (EXPRESSION) STATEMENT else STATEMENT
```

  - EXPRESSION must evaluate to a `boolean` value

```
if (i == 5) {
    // ...
} else {

}
```

Conditionals

- So far: All statements are executed in sequence
- Conditionals allow specifying a condition: If it is fulfilled, a statement is executed
- Two forms:

```
if (EXPRESSION) STATEMENT
```

```
if (EXPRESSION) STATEMENT else STATEMENT
```

  - EXPRESSION must evaluate to a `boolean` value
- The `if`-statement is a statement, therefore:

```
if (EXP1) STATEMENT else if (EXP2) STATEMENT else STATEMENT
```

is also possible

- Remember: code blocks `{ ... }` are also statements

demo

Conditional Expression

- The if-statement is a statement
- Sometimes, it's useful to make such a distinction in the form of an expression
- All other operators are unitary or binary (i.e.: take one or two values)
- Ternary operator has three parts: `EXP1 ? EXP2 : EXP3`
  - EXP1 must evaluate to a boolean value, EXP2 and EXP3 must evaluate to the same type

Conditional Expression

- The if-statement is a statement
- Sometimes, it's useful to make such a distinction in the form of an expression
- All other operators are unitary or binary (i.e.: take one or two values)
- Ternary operator has three parts: `EXP1 ? EXP2 : EXP3`
  - EXP1 must evaluate to a boolean value, EXP2 and EXP3 must evaluate to the same type
- `short daysInYear = isLeapYear() ? 366 : 365;`

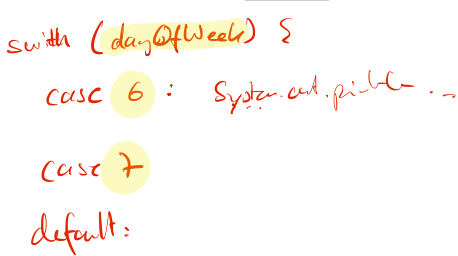
Switch-Statement

- Complex and embedded if-statements quickly become unreadable
- Alternative, if all if-statements compare against the same variable: `switch`-statement

Switch-Statement

- Complex and embedded if-statements quickly become unreadable
- Alternative, if all if-statements compare against the same variable: `switch`-statement

```
1 switch (EXPRESSION) {
2 case CONSTANT: STATEMENT; break;
3 case CONSTANT2, CONSTANT3: STATEMENT; break;
4 default: STATEMENT
5 }
```



demo

Switch-Statement

Example

```
1 static short daysInMonth(byte month) {
2     switch(month) {
3     case 2: return 28; // no break needed, because of return
4     case 4: // fall through to case 11
5     case 6:
6     case 9:
7     case 11: return 30;
8     default: return 31;
9   }
10 }
```

Section 3

Exercise