Recap: Conditionals

- Code that is executed only when conditions are met

If-Statement

true

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

Switch-Statement

```
switch (EXPRESSION) {
  case CONSTANT:
    STATEMENT;
    break;
  case CONSTANT2:
  case CONSTANT3:
    STATEMENT;
    break;
  default:
    STATEMENT
}
```

Exercise 4

UNIVERSITÄT
ZU KÖLN

Session 5: Loops

Softwaretechnologie: Java 1

Nils Reiter
nils.reiter@uni-koeln.de

November 15, 2023

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

Introduction

- Executing code repeatedly
- What do we need?
  - The code to be executed (i.e., a $\underline{\text{code block}}$) { }
  - Conditions on how often to repeat

While-Loop

- Repeat as long as some expression is true
- Similar to `if`, but with a repeat option
  - `EXPRESSION` must be of type `boolean`
  - If `EXPRESSION` evaluates to `false`, not executed at all
- `EXPRESSION` is evaluated in every iteration before the code block is run
  - I.e., if variables change during execution, the expression result may also change

```java
while (EXPRESSION) {
  // some code
}
```

false

demo

Do-While-Loop

- Repeat as long as some expression is true
- Similar to `while`, but code is executed at least once

Exp

```
do {
  // some code
} while (EXPRESSION);
```

For-Loop

- In many cases, we know in advance how often do repeat code

```
// do something for each of 25 days
int days = 25;
int c = 0;
while (c < days) {
  // do stuff
  c++; // short form of c = c + 1
}
```

For-Loop

- In many cases, we know in advance how often do repeat code

```
// do something for each of 25 days
int days = 25;
int c = 0;
while (c < days) {
  // do stuff
  c++; // short form of c = c + 1
}
```

```
// do something for each of 25 days
int days = 25;
for (int c = 0; c < days; c++) {
  // do stuff
}
```

- For-loops offer a denser notation

For-Loop

```
for (INIT; CONDITION; UPDATE) {
  //
}
```

- INIT: Executed before entering the loop for the first time
- CONDITION: An expression, checked before every iteration
  - Must be of type `boolean`
- UPDATE: Executed at the end of each iteration

For-Loop

Scope

- Variables declared within a for loop are not known outside of it
- If variables are declared in INIT, they belong to the scope of for-statement
- This shows a difference to the corresponding while-statement

Example

```
int a = 4;
for (int b = 0; b < 10; b++) {
  // b is known
  // a is known
}
// a is known
// b is not known
```

demo

Break and Continue

- All loops can *also* be controlled by two keywords: `break` and `continue`
- `break`
  - Terminates the entire loop abruptly
  - Execution continues after the closing `}`
- `continue`
  - Terminates the current iteration of the loop
  - Execution continues with the next iteration
    - `for`: Run UPDATE first
    - All loops check their conditions before

Break and Continue

- All loops can *also* be controlled by two keywords: `break` and `continue`
- `break`
  - Terminates the entire loop abruptly
  - Execution continues after the closing `}`
- `continue`
  - Terminates the current iteration of the loop
  - Execution continues with the next iteration
    - `for`: Run UPDATE first
    - All loops check their conditions before
- `break` / `continue` are sometimes useful, but
  - are able to exit a loop independently of the exit condition and thus
  - make code harder to read and understand

Understanding Loops

- Sometimes challenging to understand a loop
- Crucial: Keep track of variable contents
- Variables may change in every iteration
- Conditions/exit conditions can be complex

Understanding Loops

- Sometimes challenging to understand a loop
- Crucial: Keep track of variable contents
- Variables may change in every iteration
- Conditions/exit conditions can be complex

How many ! will be printed?

```java
int a = 7;
while(a > 0) {
  int f = a % 2;
  if (f > 0) {
    a = a - 2;
  } else {
    a = a + 1;
  }
  System.out.print("!");
}
```

Understanding Loops

- Sometimes challenging to understand a loop
- Crucial: Keep track of variable contents
- Variables may change in every iteration
- Conditions/exit conditions can be complex

How many ! will be printed?

```
1  int a = 7;
2  while(a > 0) {
3    int f = a % 2;
4    if (f > 0) {
5      a = a - 2;
6    } else {
7      a = a + 1;
8    }
9    System.out.print("!");
10 }
```

Ausgabe

Line	a	f
1	7	
2	7	
3	7	1
4	7	1
5	5	1
9	5	1
2	5	
3	5	1
4	5	1
5	3	1

!

Section 1

Exercise