Java

Tooling, variables and basic control flow

Christoph Baitis

20. Oktober 2022

Overview

1. Recap

2. Tooling

3. Warm Up

4. Deep dive into variables

5. Boolean

... and the basics of control flow

6. Loops

Recap

Recap 2/3

```
1 public class Hello {
2     public static void main(String[] args) {
3         System.out.println("Hello World!");
4     }
5 }
```

Recap 3/3

```java
import java.util.Scanner;
public class Talk {
    public static void main (String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Hi, how old are you?");
        int age = scanner.nextint();
        int age2 = age + 5;
        System.out.println("In 5 years, you'll be " + age2);
    }
}
```

Tooling

What we need...

What tools do we actually need to develop JAVA?

- texteditor (Notepad)
- compiler (javac)
- Java Virtual Machine (java)

What we want...

For easy programming we want to have

- supporting (intelligent) editor with
  - direct control of a compiler
  - auto start of the application after compilation
  - debugger (later)

What we want...

For easy programming we want to have

- supporting (intelligent) editor with
  - direct control of a compiler
  - auto start of the application after compilation
  - debugger (later)
- Visualstudio Code
  `https://code.visualstudio.com/`

What is Visual Studio Code (VS Code)

- fundamentally its just an editor
- but it can be extended with add-ons
- add-ons allow for example
  - syntax highlighting
  - code completion
  - debugging features

What is Visual Studio Code (VS Code)

- fundamentally its just an editor
- but it can be extended with add-ons
- add-ons allow for example
  - syntax highlighting
  - code completion
  - debugging features

Let's take a look at VS Code...

Warm Up

Warm Up Task

Task

- Open visual studio code and create a new file *calculator.java*
- write a class *Calculator* with the basic framework of a java application (see code below)
- read in two numbers and add them

```java
import java.util.Scanner;
public class Calculator {
    public static void main (String[] args) {
        /* add your code here and use operations like
         * + ... addition
         * - ... subtraction \
         * * ... multiplication
         * / ... devision
        */
    }
}
```

calculator.java

Deep dive into variables

Deep Dive: Variables

Java knows many different types of variables.

You should know the following variable types:

Name	Example	Definition
int	int i = 3261;	Whole numbers (-2,147,483,648 to 2,147,483,647)
float	float f = 0.420f	Floating point numbers up to 7 decimal digits
boolean	boolean b = false;	Binary state - True or False
char	char c = 'a';	Single character or ASCII code

Deep Dive: Variables

Java knows many different types of variables.

Good to know but not really important

Name	Example	Definition
byte	byte b = 11;	Whole numbers (-128 to 127)
long	long l = 31L;	Whole numbers (very big)
double	double d = 43.23d;	Like float - just twice as precise
short	short s = 423;	Whole numbers (-32,768 to 32,767)

Basic mathematical operations

You can use these basic operations when working with `int, float (, long, double, short)`

Addition	a + b;
Subtraction	a - b;
Multiplication	a * b;
Division	a / b;
Modulo	a % b;
Increment	a++;
Decrement	a–;

Try it yourself

```
1 int a = 9*4; // = 36
2 int a = 9+4; // = ??
3 int a = 9%4; // = ??
4 int a = 9/4; // = ??
```

```
1 float a = 9*4; // = ??
2 float a = 9+4; // = ??
3 float a = 9%4; // = ??
4 float a = 9/4; // = ??
```

```
1 int i = 3000 * 2;   // = ??
2 short s = 3000 * 2; // = ??
```

Task

Play around with different variable types.

What are the boundaries of the types?

Try it yourself

```
int a = 9*4; // = 36
int a = 9+4; // = ??
int a = 9%4; // = ??
int a = 9/4; // = ??
```

```
float a = 9*4; // = ??
float a = 9+4; // = ??
float a = 9%4; // = ??
float a = 9/4; // = ??
```

```
int i = 3000 * 2;   // = ??
short s = 3000 * 2; // = ??
```

Variable types have different sizes!

```
incompatible types: possible lossy conversion from int to
    short
```

Boolean
... and the basics of control flow

Boolean and boolean algebra

```
1 boolean b = true || false;
```

- With booleans, we can make logical decisions and control how our code “flows”.
- Without booleans, code would be boring and always do the exact same thing.

Boolean and boolean algebra

```
1 boolean b = true || false;
```

A boolean can only be *true* or *false*

```
1 boolean a = false;
2 boolean b = true;
```

Booleans

- What do we need booleans for?
  - to control how our program flows
  - to make decisions
- conditions are booleans

Conditions and if-statements

- What do we use conditions and if-statements for?
  - to execute different code depending on the value of the condition

```
1 if(condition) {
2     // do something cool!
3 }
```

Conditions and if-statements

- What do we use conditions and if-statements for?
  - to execute different code depending on the value of the condition

```
1 if(condition) {
2     // do something cool!
3 }
```

- conditions need to evaluate to *true* so the code inside ... is executed

```
1 if(true) {
2     // ...the code...
3 }
```

Conditions and if-statements

Conditions can be *boolean* variables

```
boolean myBoolean = true;
if(myBoolean) {
    // do something cool!
}
```

Or comparisons

```
int i;
...
if(i > 10) {
    // do something cool!
}
```

Conditions and comparisons

We can compare variables to each other using comparison operators

- the result is a *boolean*

```
1 1 < 3     // ??
2 3 > 2     // ??
3 3 <= 3    // ??
4 1 >= 1    // ??
5 1 == 1    // ??
```

Conditions and comparisons

We can compare variables to each other using comparison operators

- the result is a *boolean*

```
1 1 < 3     // ??
2 3 > 2     // ??
3 3 <= 3    // ??
4 1 >= 1    // ??
5 1 == 1    // ??
```

We also can use comparisons as conditions

```
1 int a = 3;
2 int b = 11;
3 if(a < b) {
4     System.out.println("a is smaller than b!");
5     System.out.println("Condition is true!");
6 }
```

Conditions and comparisons

We can also define else cases

```
1 int age = 12;
2 int minAge = 18;
3 if(age >= minAge) {
4     System.out.println("Come on in!");
5 } else {
6     System.out.println("You're too young.");
7 }
```

What does this program do?

Try it yourself

Task 1 [easy]

Write a program that prints a text out when a condition is *true*

Task 2 [medium]

Write a program that prints the absolute difference of two int a, int b.

a=7, b=9 $\rightarrow$ 2

a=9, b=7 $\rightarrow$ 2

Task 3 [hard]

Remember the size of different data types? e.g short and int?

Write a program that prints the product (*) of two short only if the product does not exceed the limit of short (32,767) only using variables of type short.

Loops

Loops

- Loops let us execute the same code multiple times
- Loops continue as long as a condition is true ("satisfied")
- Java has two general types of loops: while and for

Loops

A *while* loop is the easiest

.. do something while (as long as) a condition is satisfied

```
1 boolean myLoopCondition = true;
2 while(myLoopCondition) {
3     // this section will get executed multiple times
4 }
```

Question: How long will this loop continue for?

Loops

A *while* loop is the easiest

.. do something while (as long as) a condition is satisfied

```
1 boolean myLoopCondition = false;
2 while(myLoopCondition) {
3     // this section will get executed multiple times
4 }
```

Question: How long will this loop continue for?

Loops

How do we avoid infinite loops?

$\rightarrow$ We can use variables to dynamically change our loop condition once we want to

```
1 int a = 0;
2 while(a < 10) {
3     a = a+1; // increment a
4     System.out.println(a);
5 }
```

Question: What happens here?

Loops

With continue and break we can escape a loop or skip an iteration

```
int a = 0;
while(true) {
    if(a == 10) {
        break;
    }
    a++;
}
```

Question: What happens here?

Try it yourself - the final task

Your first JAVA game!

Write a game which first calculates a random number and lets the user guess the number afterwards.

The user should be promoted to enter a number

- if the number is larger than the random number the program should output *"To large!*
- if the number is small than the random number the program should output *"To small!*
- if the number is the random number the program should output *"You got it!*
- also output the number of tires the user took to guess the number

```
1 //Generating a random number between a min and a max value
2 int min = 5
3 int max = 10;
4 int random = ((Math.random() * (max - min)) + min);
```

That's it!

- Be encouraged to keep working on the tasks
- Feel free to reach out
  - to send your results
  - to tell me about problems you ran into

Next lesson

- for-loops
- Objects
- classes & methods
- more practical examples!

Self study

For those of you who are bored because you can finish the tasks much quicker than others challenge yourself!

You can find other tasks at
`https://ein-christoph.github.io/java-tud/self-study.html`

Caution!
The tasks you'll be confronted with in INLOOP are for people already knowing the basics of Java. Do not worry if you can not solve the tasks right now. You will be able to after this course!