Java Introduction

Wojciech Frącz
fracz@agh.edu.pl
AGH IEiT

Resources

- Moodle: https://moodle.ki.agh.edu.pl
- Course name: EFREI - Java 1
- Password change: https://accounts.ki.agh.edu.pl (Google Translate is your friend)
- Short test before each lab

What is Java?

Why Java?

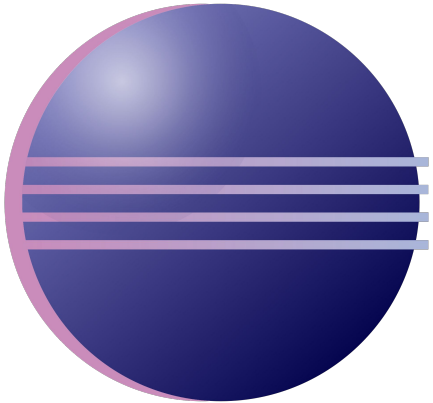
Java

- High level language
- Object-Oriented Programming (OOP)
- Simple syntax
- Portability
- Many ready-to-use solutions in standard library

HelloJava.java

```java
public class HelloJava {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Eclipse



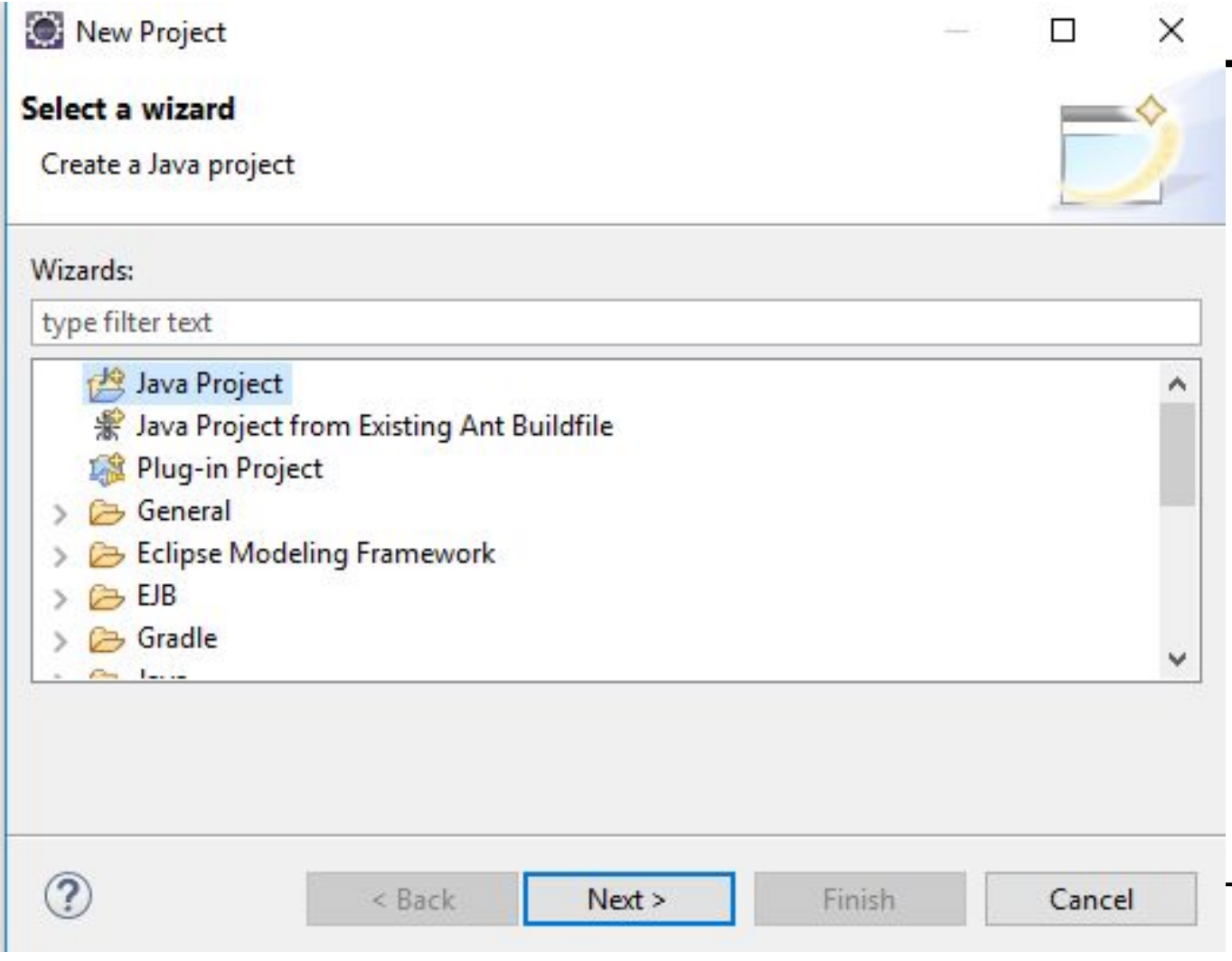
- IDE - Integrated Development Environment
- compiles, launches, facilitates writing of a source code
- http://www.eclipse.org
- `C:\eclipse\eclipse-neon`

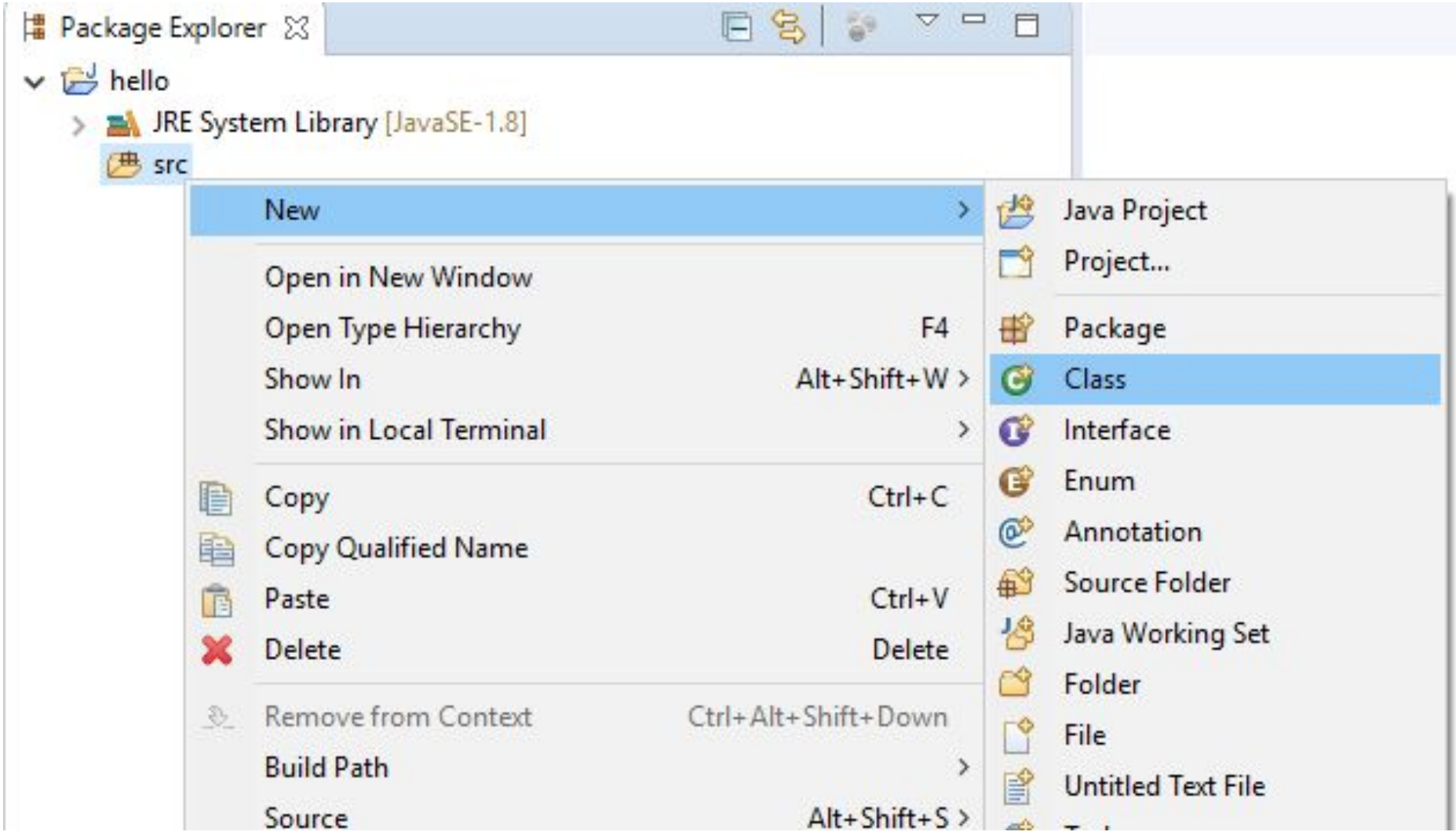
eclipse-workspace - Eclipse

File Edit Navigate Search Project Run Window Help

- New Alt+Shift+N >
- Open File...
- Open Projects from File System...
- Close Ctrl+W
- Close All Ctrl+Shift+W
- Save Ctrl+S
- Save As...
- Save All Ctrl+Shift+S
- Revert

- Maven Project
- Enterprise Application Project
- Dynamic Web Project
- EJB Project
- Connector Project
- Application Client Project
- Static Web Project
- JPA Project
- Project...
- Servlet


New Project
Select a wizard
Create a Java project
Wizards:
type filter text
Java Project
Java Project from Existing Ant Buildfile
Plug-in Project
General
Eclipse Modeling Framework
EJB
Gradle
< Back
Next >
Finish
Cancel


Package Explorer
hello
JRE System Library [JavaSE-1.8]
src
New
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File

New Java Class

Java Class

Create a new Java class.

Source folder: hello/src [Browse...]

Package: hello [Browse...]

☐ Enclosing type: [Browse...]

Name: HelloJava

Modifiers: ◉ public ○ package ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object [Browse...]

Interfaces: [Add...]

Which method stubs would you like to create?

☑ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

[Finish] [Cancel]

- hello
  - JRE System Library [JavaSE-1.8]
  - src
    - hello
      - HelloJava.java

```
1 package hello;
2
3 public class HelloJava {

    public static void main(String[] args) {
        System.out.println("Hello Java");
    }
```

New		
Open	F3	
Open With		
Open Type Hierarchy	F4	
Show In	Alt+Shift+W	
Show in Local Terminal		
Copy	Ctrl+C	
Copy Qualified Name		
Paste	Ctrl+V	
Delete	Delete	
Remove from Context	Ctrl+Alt+Shift+Down	
Build Path		
Source	Alt+Shift+S	
Refactor	Alt+Shift+T	
Import...		
Export...		
References		
Declarations		
Refresh	F5	
Assign Working Sets...		
Coverage As		
Run As		1 Run on Server — Alt+Shift+X, R
		2 Java Application — Alt+Shift+X, J
		Run Configurations...
Debug As		
Profile As		
Validate		

“From Oak to Coffee”

Fast Java history

1991 - Sun starts a *Green* project.
One source code, multiple platforms. New language called *Oak*

1996 - Java 1.0 available, basic browsers support.

1998 - Java 2

...
2009 - *Oracle* buys *Sun*

2017 - Java 9

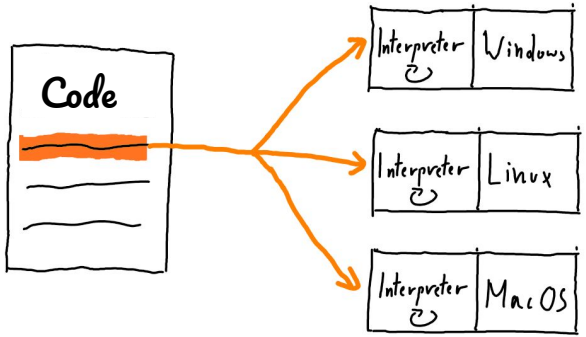
2018 - Java 10.3

Java
Architecture

Compilers vs Interpreters

Interpreter

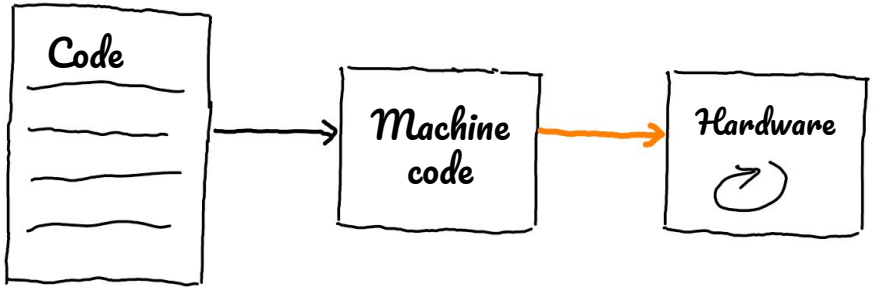
- Analyses the source code
- **Executes it directly** when launched
- Generally slower
- Flexible, comfortable for prototyping, testing, debugging



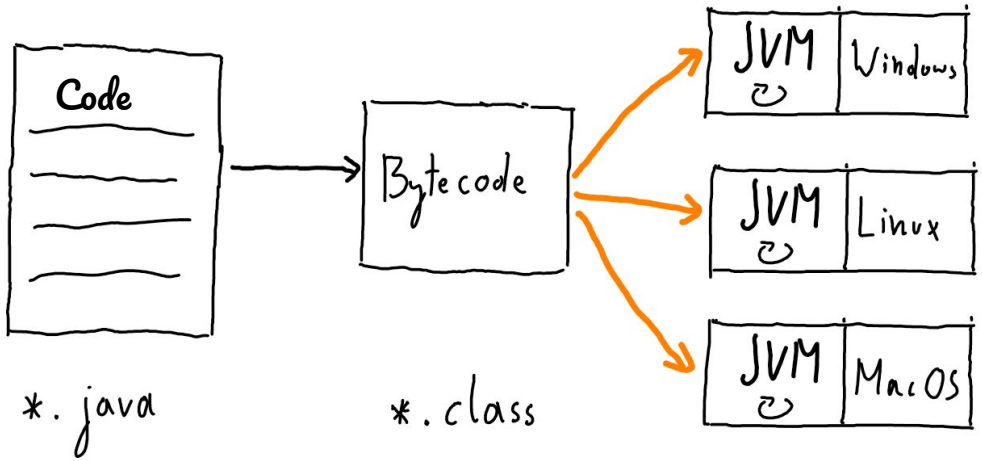
Compilers vs Interpreters

Compiler

- Translates the source code to another form (often: machine code)
- **Does not execute the source code**
- Can optimize it during the translation
- The machine can be executed faster, but the machine code is hardware-specific



JVM - Java Virtual Machine



- **Compilation** into an intermediate form (*bytecode*)
- **Interpretation** of the *bytecode* in JVM (different implementations for different platforms)

JVM - Java Virtual Machine

- Portability: *“Write Once, Run Everywhere”*
- For casual users: JRE
- Also: other languages can be launched in JVM:
  - Scala
  - Clojure
  - Kotlin
  - Jython (Python), JRuby (Ruby), itp.

Java Development Kit (JDK)

javac : file**.java** → file**.class**
(compilation)

java : file**.class** → JVM (interpretation)

Java
Syntax

What does a program consist of?

"Algorithms + Data structures = Programs"

- N. Wirth

Instructions

Data of different types
(variables, constants)

Data types in Java

Built-in scalars **(primitive values)**:

- int - an integer
- double - a floating point number
- boolean - a logical value
- char - a character

```
12 -10 0

3.14 1.0 2e10

true false

'a' 'z' 'ę'
```

Variables

Variable

- Represents data of a declared type
- Type + variable name is a **declaration**

```
int shoeSize;
```

Data type

Variable name

Variables

Variable

- Represents data of a declared type
- Type + variable name is a **declaration**
- Stores a value
- The first assignment is sometimes called as **initialization**

```
int shoeSize = 40;
```

Data type — Variable name — Data value

Variables

Variable

- Represents data of a declared type
- Type + variable name is a **declaration**
- Stores a value
- The first assignment is sometimes called as **initialization**
- The value can be changed

```
int shoeSize = 40;

shoeSize = 41;
```

Variables

What can you do with a scalar variable?

- Assign a value

```
int number = 149;
```

*"After that line, the **number** variable has an 149 value"*

ATTENTION: Equal sign does not mean "compare" but "assign"!

Variables

What can you do with a scalar variable?

- Assign a value
- Assign a value from other variable

```
double yourResult = 3.14;

double myResult = yourResult;
```

*"After that line, the **myResult** variable has a 3.14 value."*

Variables

What can you do with a scalar variable?

- Assign a value
- Assign a value from other variable
- Use it with an operator (e.g. **arithmetic**)

```
int year = 2018;

year = year + 1;
```

*"After that line, the **year** variable has a 2019 value"*

Variables

What can you do with a scalar variable?

- Assign a value
- Assign a value from other variable
- Use it with an operator (e.g. **logic**)

```
int potatosWeight = 2;
```

```
potatosWeight > 1
```

“Do potatos weigh more than 1kg?"

```
potatosWeight == 2
```

“Do potatos weigh exactly 2kg?”

Task 1

Print sum of variables `a` and `b`.

```
public class HelloJava {
    public static void main(String[] args) {
        int a = 1;
        int b = 4;
    }
}
```

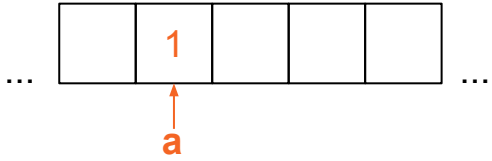
Quiz 1

What is the output of the following program?

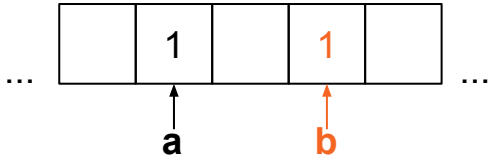
```
public class HelloJava {
    public static void main(String[] args) {
        int a = 1;
        int b = a;
        a = 2;
        System.out.println(b);
    }
}
```

Primitive types assignments

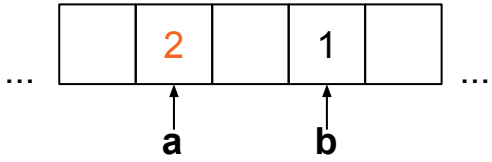
```
int a = 1;
```




```
int b = a;
```




```
a = a + 1;
```



In primitive types, the **value** is being assigned.

Data types in Java

- Java is a **statically typed** language!
- It means that every variable must have a declared type and it cannot be changed over time
- Compiler checks the correctness of the operations regarding their type

Python (dynamically typed):

```python
number = 100
```

Java (statically typed):

```java
int number = 100;
```

Data types in Java

✔ `int a = 40;`

✘ `int b = 2.5;` → `int b = (int)2.5;`

✔ `double c = 3;` (lossless casting)

✘ `boolean d = 1;` → `boolean d = true;`

✔ `boolean e = a == 40;` (== operator produces **true** or **false**)

Task 2

Use Java to solve the equation below for x=2 and y=0.5

```
(2x + 3y) * 5x / 2
```

Classes

Complex types = **Classes**

- consist of attributes (fields) and functions (methods)
- allow to define your own data types
- introduce object-oriented programming!

- every Java program contain multiple classes that work with each other to deliver the desired functionality

Classes

Classes (examples):

- `String`
- `System`
- `HelloJava`

  …
- `Car`
- `Instrument`

String

`String` class

- Represents a sequence of characters (text)

```
String song = "I'm happy and I know it";
```

Data types in Java

`String` class

- Represents a sequence of characters (text)
- Implements some operations (methods)

```
String song = "I'm happy and I know it";
int songSize = song.length();
```

You call method on an object with a dot

Data types in Java

`String` class

- Represents a sequence of characters (text)
- Implements some operations (methods)

```
String song = "I'm happy and I know it";
int songSize = song.length();
String mood = song.substring(4, 10);
```

Task 3

Print *Hello World* using the declared variables.

```
public class HelloJava {
    public static void main(String[] args) {
        String hello = "Hello";
        String world = "World";
    }
}
```

Arrays

Another complex type : **Array**

- Sequence of elements of the same type

```
int[] lotteryResult;
```

Array of integers

Arrays

Another complex type : **Array**

- Sequence of elements of the same type
- Has a fixed size chosen during initialization

```
int[] lotteryResult = new int[6];
```

Array of integers

Arrays

Variable →

Array →

Arrays are 0-indexed!

```
int[] lotteryResult = new int[6];
lotteryResult[0] = 5;
int secondNumber = lotteryResult[1];
// ...
lotteryResult[5] = 22;
// lotteryResult[6] = 77;
```

This is a comment, BTW

https://www.mistickers.pl/real-programmers-count-from-0-kubek-dla-programisty.html

Instructions (Control Flow Statements)

- Conditionals
- Loops

Variables

What can you do with a scalar variable?

- Assign a value
- Assign a value from other variable
- Use it with an operator (e.g. **logic**)

```
int potatosWeight = 2;
```

```
potatosWeight > 1
```

“Do potatos weigh more than 1kg?"

```
potatosWeight == 2
```

“Do potatos weigh exactly 2kg?”

Conditional instruction: `if`

```
if (potatosWeight < 2) {
   System.out.println("I will do it!");
}
else {
   System.out.println("No way");
}
```

Conditional instruction: `if`

```
if (potatosWeight < 2) {
   System.out.println("I will do it!");
}
else if (potatosWeight < 5) {
   System.out.println("Have to go to a gym first");
}
else {
   System.out.println("No way");
}
```

Task 4

Write a program that will declare a `grade` variable and depending on it:

- prints "It's cool!" if `grade` is 5
- prints "It's nice" if `grade` is less than 5 greater than or equal 3
- prints "I must start learning" if `grade` is less than 3 but greater than or equal 2
- prints "Something is wrong" in every other case

Program input & output (I/O)

```
Scanner in = new Scanner(System.in);
System.out.println("What's your name?");
String name = in.nextLine();
System.out.println("How old are you?");
int age = in.nextInt();
System.out.print("Hello " + name + "! You are so ");
if (age > 50) {
  System.out.println("old!");
} else {
  System.out.println("young!");
}
```

Program input & output (I/O)

Create a handler for Input

```
Scanner in = new Scanner(System.in);
System.out.println("What's your name?");
String name = in.nextLine();
System.out.println("How old are you?");
int age = in.nextInt();
System.out.print("Hello " + name + "! You are so ");
if (age > 50) {
  System.out.println("old!");
} else {
  System.out.println("young!");
}
```

Program input & output (I/O)

Print some output with newline character at the end

```
Scanner in = new Scanner(System.in);
System.out.println("What's your name?");
String name = in.nextLine();
System.out.println("How old are you?");
int age = in.nextInt();
System.out.print("Hello " + name + "! You are so ");
if (age > 50) {
  System.out.println("old!");
} else {
  System.out.println("young!");
}
```

Program input & output (I/O)

```
Scanner in = new Scanner(System.in);
System.out.println("What's your name?");
String name = in.nextLine();
System.out.println("How old are you?");
int age = in.nextInt();
System.out.print("Hello " + name + "! You are so ");
if (age > 50) {
  System.out.println("old!");
} else {
  System.out.println("young!");
}
```

Read one line from the input (from the user)

Program input & output (I/O)

```
Scanner in = new Scanner(System.in);
System.out.println("What's your name?");
String name = in.nextLine();
System.out.println("How old are you?");
int age = in.nextInt();
System.out.print("Hello " + name + "! You are so ");
if (age > 50) {
  System.out.println("old!");
} else {
  System.out.println("young!");
}
```

Read one integer from the input (from the user)

Program input & output (I/O)

```
Scanner in = new Scanner(System.in);
System.out.println("What's your name?");
String name = in.nextLine();
System.out.println("How old are you?");
int age = in.nextInt();
System.out.print("Hello " + name + "! You are so ");
if (age > 50) {
  System.out.println("old!");
} else {
  System.out.println("young!");
}
```

Print a greeting without a newline character at the end

Program input & output (I/O)

```
Scanner in = new Scanner(System.in);
System.out.println("What's your name?");
String name = in.nextLine();
System.out.println("How old are you?");
int age = in.nextInt();
System.out.print("Hello " + name + "! You are so ");
if (age > 50) {
  System.out.println("old!");
} else {
  System.out.println("young!");
}
```

Finish execution by printing appropriate suffix and a newline.

Loops

- execute code many times
- you can alter their behavior in every loop by applying modifications to variables they operate on

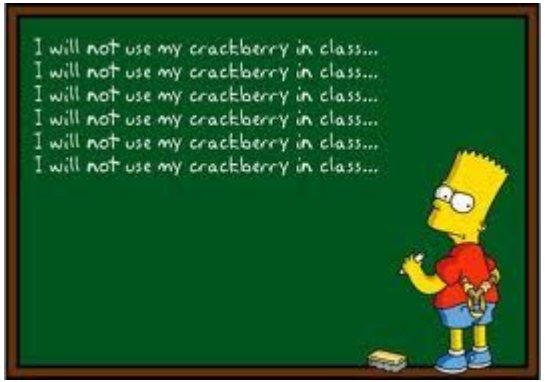
while

```
int donePushups = 0;
while (donePushups < 10) {
    System.out.println("Doin a pushup!");
    donePushups += 1;
}
```

for

```
for (int donePushups = 0; donePushups < 10; donePushups++) {
    System.out.println("Doin a pushup!");
}
```

Task 5



Imagine that your brother received a penalty task from the teacher. He has to write "I will not learn programming on a history lesson." 100 times!

Help him, writing a program with a loop.

Take into consideration that the teacher looks only at the first and the last ten lines when verifying the homework. You should write "But why? Programming is cool!" starting on line 11 and ending on line 89.

What's next?

- Take a look at Java primitive types list https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html
- Java operators https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html
- Instructions https://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html
- Make sure you understand everything from the slides :-)