90 Minutes of Java Syntax

Jakub Waszczuk

Heinrich Heine Universität Düsseldorf

Winter Semester 2018/19

Goal

At the end of this session, you should:

- become/remain familiar with Java syntax
- be able to compile and execute Java code

Goal

At the end of this session, you should:

- become/remain familiar with Java syntax
- be able to compile and execute Java code

What this session does not aim at:

- provide an introduction to algorithmics
- introduce complicated object oriented programming concepts

Getting started

First, we need to install the Java Development Kit

Ubuntu (OpenJDK)

sudo apt install default-jdk

Windows, Mac OS

`http://www.oracle.com/technetwork/java/javase/downloads/`

Getting started

First, we need to install the Java Development Kit

Ubuntu (OpenJDK)

```
sudo apt install default-jdk
```

Windows, Mac OS

`http://www.oracle.com/technetwork/java/javase/downloads/`

Good news: you might actually have to do this step right now (in case of problems, online compilers such as `https://www.jdoodle.com/online-java-compiler` are available)

Getting started

For large scale programs, using an **Integrated Development Environment** can be helpful

Ubuntu (obsolete?)

sudo apt install eclipse

Ubuntu (worked for me)

sudo apt install snapd
sudo snap install --classic eclipse

Windows, Mac OS

`http://www.eclipse.org/downloads/packages/release/2018-09/r/eclipse-ide-java-developers`

Useful commands (if not using Eclipse)

Compile a file

```
javac a_file.java
```

Execute a program

```
java program
```

(if the file `program.class` was created by compilation)

Create a simple project (with Eclipse)

- File → New → Java Project
- enter a project name and click 'Finish'
- right click on the project (in the package explorer on the left)
- New → Class
- enter a name
- tick the box

  `public static void main ...`

  and click 'Finish'

What was that?

```
public class Test {

  public static void main(String[] args) {

  }

}
```

- a special class (you are not supposed to create objects of class `Test`)
- the entry point of the program (execution: `java Test`)

The highly original first program

```
public class Test {

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

}
```

- `println` is a built in function (provided by the `PrintStream` class)
- it takes as argument different types of values

The highly original first program

```
public class Test {

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

}
```

- `println` is a built in function (provided by the `PrintStream` class)
- it takes as argument different types of values
- how do I check which types of values are possible?
- how important are types in Java?

Java Documentation

Your new favorite webpage

`http://docs.oracle.com/javase/8/docs/api/`

Eclipse also gives some documentation when you let the pointer on keywords.

Types in Java

- when is it necessary to specify types?

Types in Java

- when is it necessary to specify types? All the time

Types in Java

- when is it necessary to specify types? All the time
- built in types are as usual: `int`, `boolean`, `char`, `(String)`...
- new object types are created by writing new classes

```
boolean b=false;
char c= 'a';
String s="This is a string";
int n=1+3;
```

Data structures in Java

- Java offers a lot of different data structures (see documentation), let us start with a basic one: arrays
- arrays are declared by giving the type of the objects used it: `type[] a`
- they are initialized by giving a length: `a= new type[length]`

```
int[] array= new int [4];
```

Conditions in Java

```
if(condition){
  set_of_instructions
}
else{
  set_of_instructions
}
```

- condition is a boolean expressions: useful operators are

  `&&, ||, >, <, >=, <=, ==, !=`

Loops in Java (`while`)

```
while(stop_condition){
  set_of_instructions
}
```

- `stop_condition` is a boolean expression
- do not forget to increment the loop index if you use one

Loops in Java (`for`)

```
for(variable_initialization; stop_condition;
    increment_expression){
  set_of_instructions
}
```

- `variable_initialization` is either creating and initializing a new variable (local to the loop) or using an existing one
- `stop_condition` is a boolean expression
- `increment_expression` explains what to change before the next iteration of the loop
- a typical loop (useful for arrays for example):

```
for(int i=0; i<t.length; i++)
```

More data structures in Java: Maps and Sets

- In these sessions, we will make intensive use of maps (association tables) and sets
- You first need to import the needed module

```
import java.util.HashMap;
```

- and instantiate objects of type HashMap (or HashSet)

```
HashMap<String, Integer> mymap=
                new HashMap<String, Integer>();
mymap.put("Bob",32);
if (mymap.containsKey("Bob")) {
  int avar = mymap.get("Bob");}
```

- https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html
- https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html

Subprograms (procedures)

```
public/private static void name (type1 arg1, type2 arg2...){
  set_of_instructions
}
```

- **public/private** determines in which class it is possible to call this procedure
- **static** says that this procedure belongs to the class, not to a specific instance
- **void** is a type (a pretty strange one)

```
public static void main (String args[])
```

- to execute a procedure:

```
name(arg1, arg2...);
```

Subprograms (functions)

```
public/private static type name (type1 arg1, type2 arg2...){
  set_of_instructions
  return value
}
```

- exactly the same except the return statement
- value must be of type type

```
public static int mult(int x, int y){
  int r=0;
  for(int i=0; i<y; i++){
    r=r+x;
  }
  return r;
}
```

- using a function: anywhere you need a value of type type

```
int r=mult(4,9);
```

Conclusion

- a program must have a `main` function
- everything has to be explicitely typed
- Java comes with a lot of useful libraries
- `http://docs.oracle.com/javase/8/docs/api/`