Foundations

Stack → a data structure implementing **pop** from a list, **pushing** the popped element to another list.

Functional programming → describes the input processing method
Imperative programming → describes the exact command the machine executes

Java

In Java, we compile code that gets fed to the **Java Virtual Machine**, which has got a different machine code from the actual machine.
It is also possible to **simulate multiple processors** on a JVM that runs on a single core.
This is because Java is **meant to be machine-independent.**

`javac` is the command to call the compiler, needed to compile the file.
After compiling a **.class file** gets produced.
The **object code** is what is produced after compilation, that must be run by calling `java` + the filename without the extension (`.class`).
However, not every program needs an object to run.
In IDEs, `src` (source) normally contains the code.
Each project can have different programs that can have different pieces of code. The actual program that can be run is the piece of code containing the **"main"** method. It's the method looked for by the compiler.

Classes

Normally stored one for a file (**that must have the same name as the class**), their syntax is

```
pubic class <<classname>> {
    body
}
```

It is possible to write public static void main with psvm; it is possible to write, in a class:
1. Declaration of an attribute (passive object that store info)
2. Declaration of a method (active objects)
3. Declaration of the main method → entry point

All of these are members of the class. It is also possible to have constructors, inner classes (classes inside classes), local classes (classes inside methods). Don't confuse members with objects of that class! The most important is the main, needed to run programs. It is possible to have more than one main. The order of the methods does not count.

Methods can have a type (public static <<type>>), receive input and outputs (return ()). Note that inputs need type. Since main has no return type, its type is void. However, it's possible to put a return with no argument in void functions. When creating a variable, we create a space for that and put a value. If we refer to that value it does not change the original; the modified version gets destroyed with the object. This is pass by value.

Note that classes are both data structures and capable of functionalities.

The main method has a String[] array called args as input. All args are necessary for the call.

It is not possible to leave a variable/attribute uninitialized.

Static means that we are not dealing with an object at all; they can be invoked without any attached object.

Difference between attributes and variables:

While

Syntax:

```
While (cond.) {
}
```

In Java, it is possible to write ++ and -- next int variable names, in order to change them by a single unit.

Difference between putting ++ before or after the var name:

To begin with, an assignment is also an expression, returning the value being assigned.
In ++x, the return value is x+1; in x++ the return value is x.

Useful commands:

- System.out.println() → note that this is a method of the System.out object. Expression can be evaluated in here.

- "" for strings
- /* multi-line comment */
- // single-line comment