

Intro to Java

Warm-up

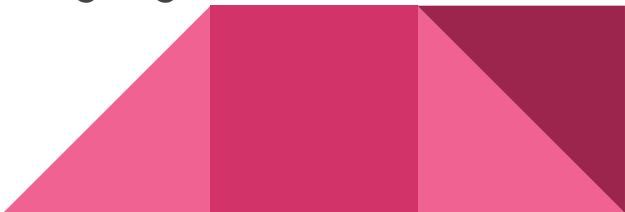
Computers can understand and run only binary code.

Programmers use high-level programming languages (Python, Java, JavaScript).

So, how can computers run the code written in a high level language?

Compilers and **Interpreters** can do the trick.

Languages whose programs we usually compile are called compiled languages.
Similarly, those we usually interpret are called interpreted languages.



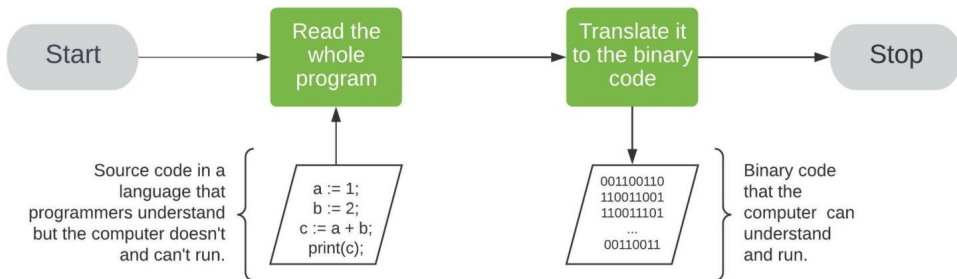
Compilers

Compilers take a whole program as input and translate it to an executable binary code in several steps.

We can run the binary code only on the machine on which we compiled it. That's because the binary code depends on the hardware and is not portable.

The compilation step is required only once. Afterward, we can run the binary code as many times as we want.

Example: C++



Interpreters

Interpreters read and execute the program at hand instruction by instruction. After being read, each instruction is translated into the machine's binary code and run.

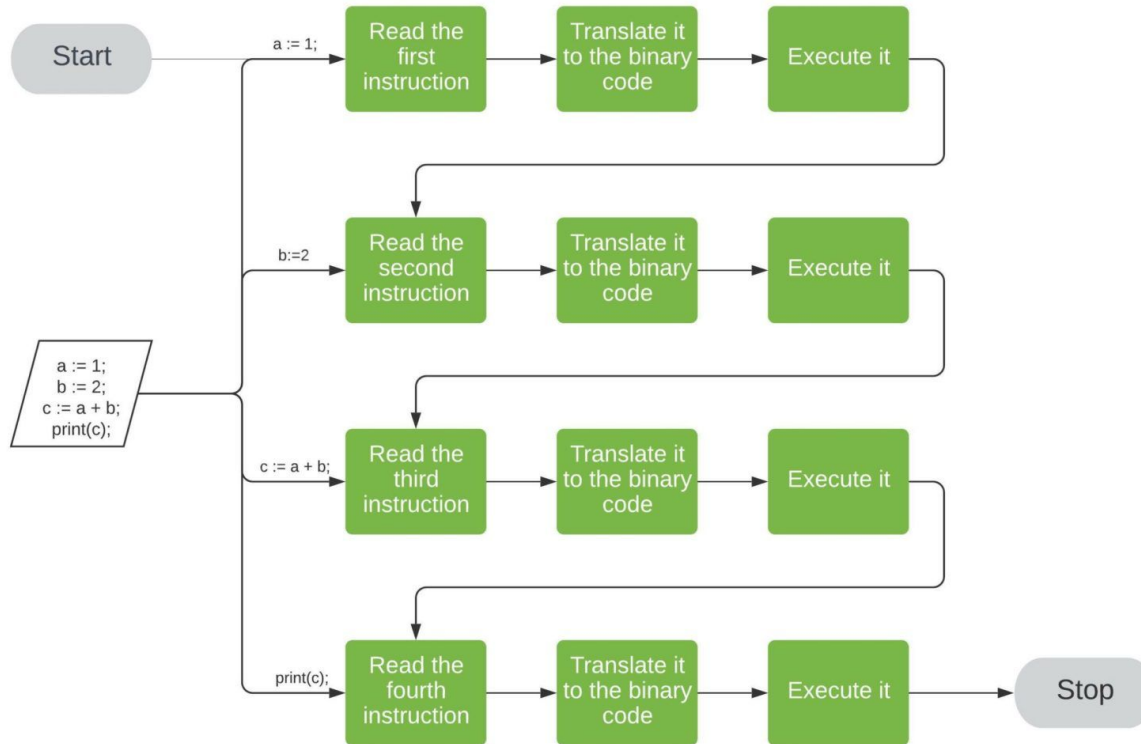
Unlike compilers, the interpreters do not produce a binary executable file. Each time we run a program, we invoke the interpreter. It then reads and executes the program one instruction at a time.

That's why it must be present in the computer's RAM whenever we run a program. In contrast to interpreters, we need compilers only during compilation.

Example: Python



Interpreters



Is Java Compiled or Interpreted?

Java is a **hybrid** of compiled and interpreted language. Java programs **compile** into java **byte code**, that does not run directly on a computer, they go through the java virtual machine (JVM)

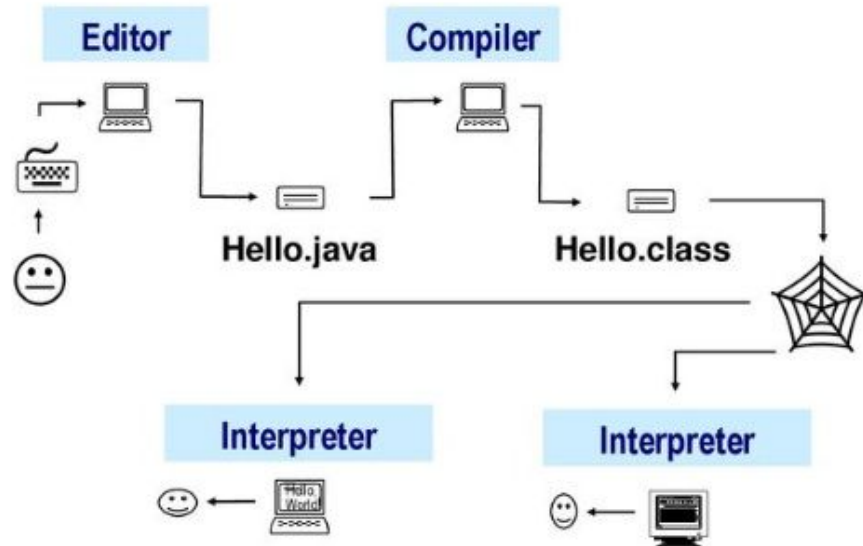
The **JVM** is like an **interpreter**. It translates from byte code to machine instructions. Think of the byte code as partially translated to machine language but not completely.

The reason to do things this way: It is faster to interpret this byte code.



How JAVA works?

1. Write Java code and save the file with **.java**
2. Now, this file will be compiled using the Java compiler, which is **javac**.
3. The Java Compiler will compile the Java file and create a .class file having **byte code** (which is not actually a machine code, unlike the C compiler)
4. This generated **byte code** is a **non-executable** code, and now it needs an **interpreter** to convert it into **machine code**. Here the JVM handles it.
5. Now, JVM will execute this byte code to execute Java byte code on a machine.
6. Now, our program will perform the functionality and gives the desired output.



Starting with Java 11 and for the first time in the programming language history, you can execute a script containing Java code directly without compilation.

The source code is compiled in memory and then executed by the interpreter, without producing a .class file on disk.

First Java Program

Every program in Java is written as a class. Java is an object-oriented language and we'll learn more about classes and objects later. Inside the class, there can be a main method that starts the program. When you ask the Java run-time to run a class, it will always start execution in the main method. Here is the template for a simple Java program with a main method:

```
public class MyClass
{
    public static void main(String[] args)
    {
        // Put your code here!
    }
}
```

Class Names should be in **UpperCamelCase**.

Try to use nouns because a class is normally representing something in the real world.



Print Methods

Java has two different methods to print output to the screen:

- `System.out.println(value)` : prints the value followed by a new line
- `System.out.print(value)` : prints the value without advancing to the next line



Comment Formats

Block Comments

Block comments are used to provide descriptions of files, methods, data structures and algorithms. Block comments may be used at the beginning of each file and before each method. They can also be used in other places, such as within methods. Block comments inside a function or method should be indented to the same level as the code they describe.

```
/*  
 * Here is a block comment.  
 */
```

Single-Line Comments

Short comments can appear on a single line indented to the level of the code that follows. If a comment can't be written in a single line, it should follow the block comment format

```
if (condition) { /* Handle the condition. */
```

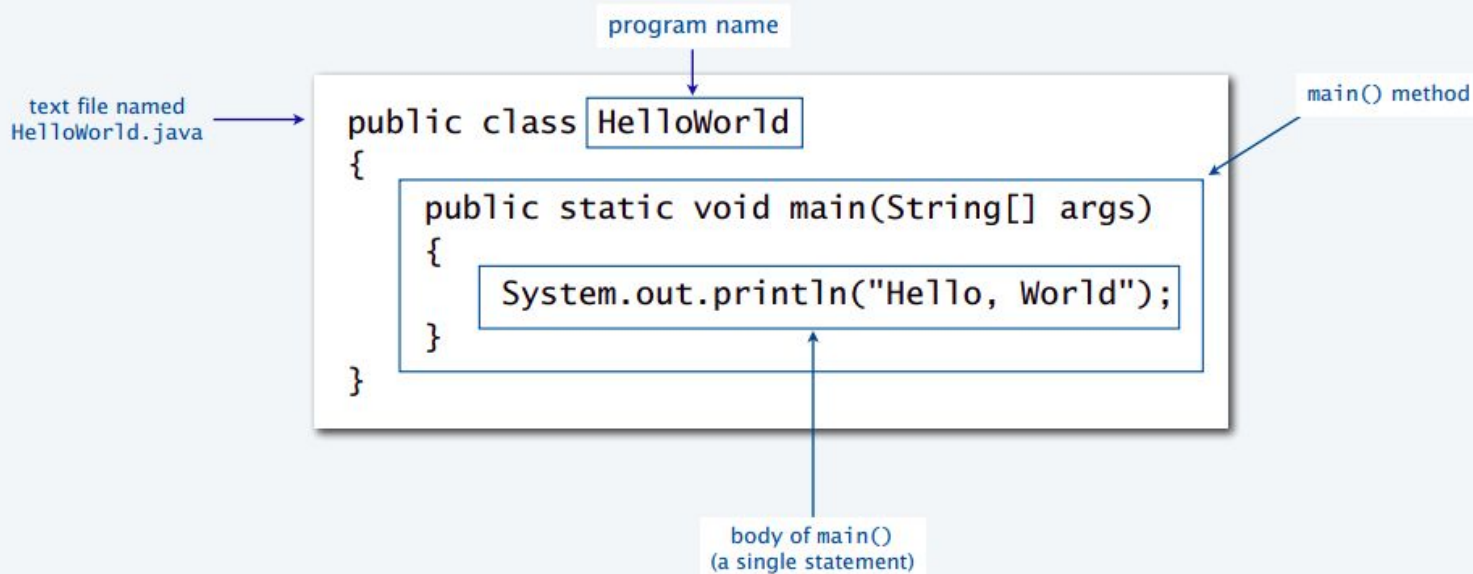
```
...  
}
```

```
if (condition) { // Handle the condition
```

```
...  
}
```



Anatomy of your first program



Exercise

What do you think is the reason to get the following errors?

```
% javac MyProgram.java
% java MyProgram
Main method not public.
```

```
% javac MyProgram.java
MyProgram.java:3: invalid method declaration; return type required
    public static main(String[] args)
                   ^
```

Solution

```
% javac MyProgram.java  
% java MyProgram  
Main method not public.
```

Solution: Must have forgotten "**public**"

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

```
% javac MyProgram.java  
MyProgram.java:3: invalid method declaration; return type required  
    public static main(String[] args)  
                   ^
```

Solution: Aha! Forgot "**void**"

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

Developing your programs in Java

1. EDIT your program

- Create it by typing on your computer's keyboard.
- Result: a text file such as HelloWorld.java.

2. COMPILE it to create an executable file

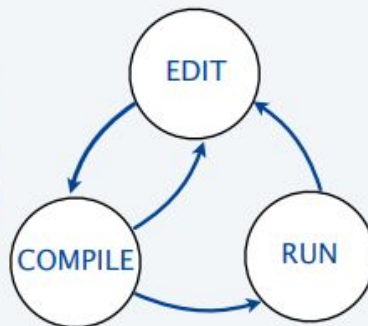
- Use the Java compiler
- Result: a Java bytecode file such as HelloWorld.class
- Mistake? Go back to 1. to fix and recompile.

not a legal Java program

3. RUN your program

- Use the Java runtime.
- Result: your program's output.
- Mistake? Go back to 1. to fix, recompile, and run.

a legal Java program that does the wrong thing



Compile-time Errors

Compile-time errors are detected by the Java compiler before the program runs. These include syntax errors such as missing symbols, type mismatches.



Runtime Errors

Runtime errors occur:

- After the program compiles successfully
- While the program is executing
- Examples: dividing by zero, accessing invalid indexes, null references

