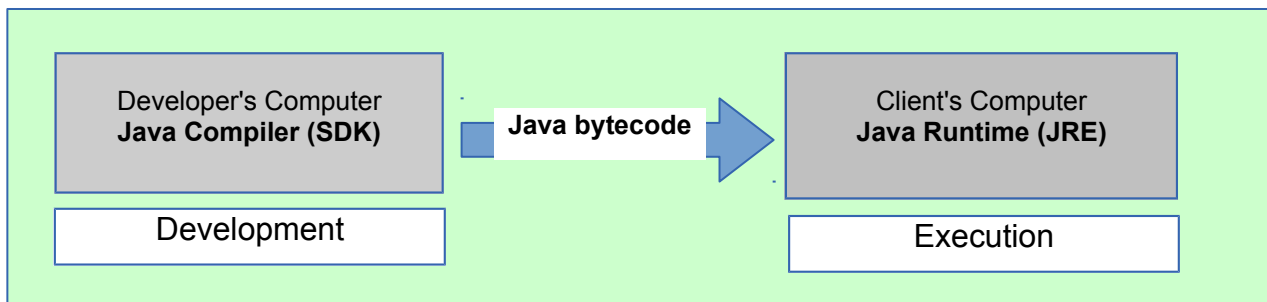


Principles of Programming with Java

Java is a *platform independent* programming language. This important attribute was introduced by the separation of the development from the execution process through the inception of intermediate code (bytecode).

- On the developer's system Java sourcecode is compiled to bytecode using the Java SDK (Software Development Kit).
- Java bytecode is transferable and can be executed on any client machine and any combination of operating system and processor type as long as there is present a locally installed Java JRE (Java Runtime Environment).



So it is getting clear that Java **bytecode** plays a central role in the overall programming system. Bytecode may be regarded as a special assembler code which contains instructions (microcode) for the Java Runtime Environment. It contains the hardware-independent representation of the original Java sourcecode. This intermediate code is suitable for transmission in network environments because of its pure binary form. In a certain sense the Java Runtime Environment may be depicted as a <CPU in software> for executing bytecode. Bytecode can be decompiled so that the original source code may be noticed.

The multistep development process (compiling-transferring-testing/executing) obviously complicates Java programming altogether – at least compared to languages such as C or C++, although the use of an integrated development environment (IDE) as Eclipse and NetBeans facilitates and summarizes the necessary steps.

On the other hand this multistep development process exactly meets today's demands for distributed applications on the Internet (client-server model). The server is the host machine that provides a service (or program), the client (customer) is the machine who is able to use/execute the service.

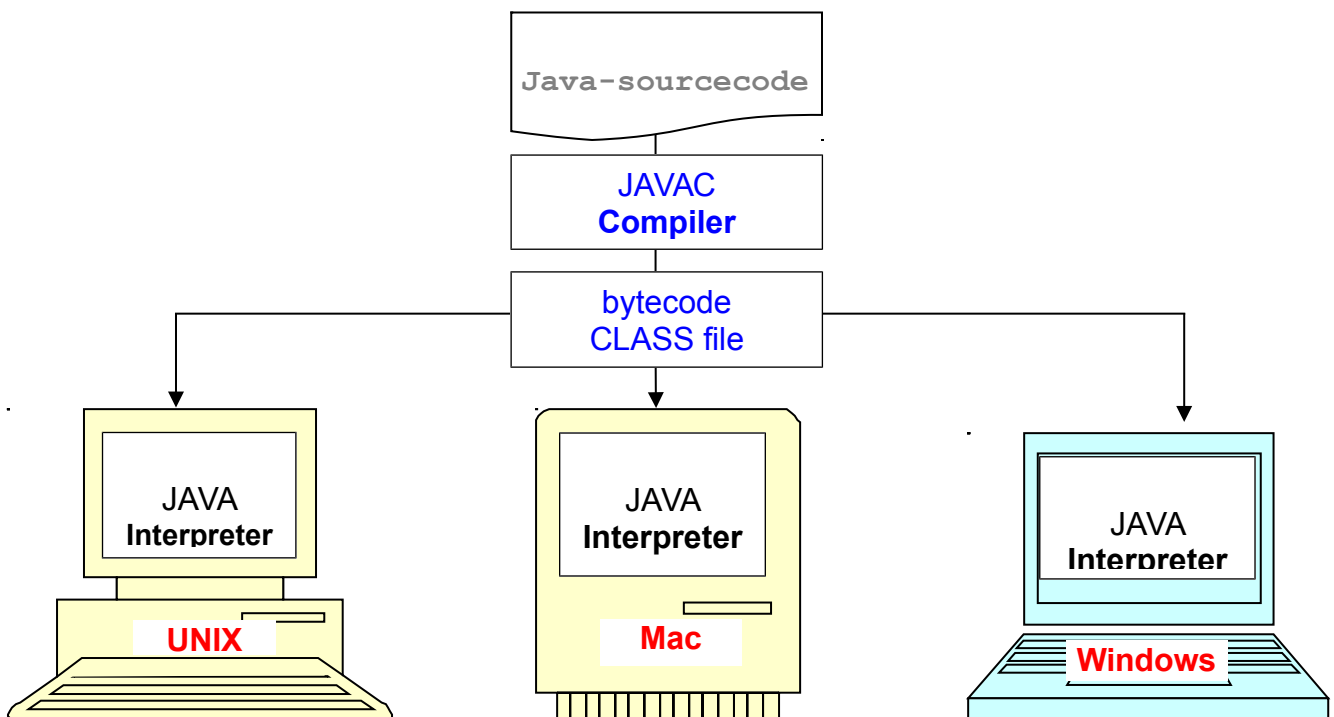
Two different Java systems are offered by Java manufacturers (Oracle, OpenJDK etc.)

- SDK for development and testing Java bytecode (includes Compiler and Runtime)
- JRE for running Java bytecode (only Runtime)

To summarize the Java development process:

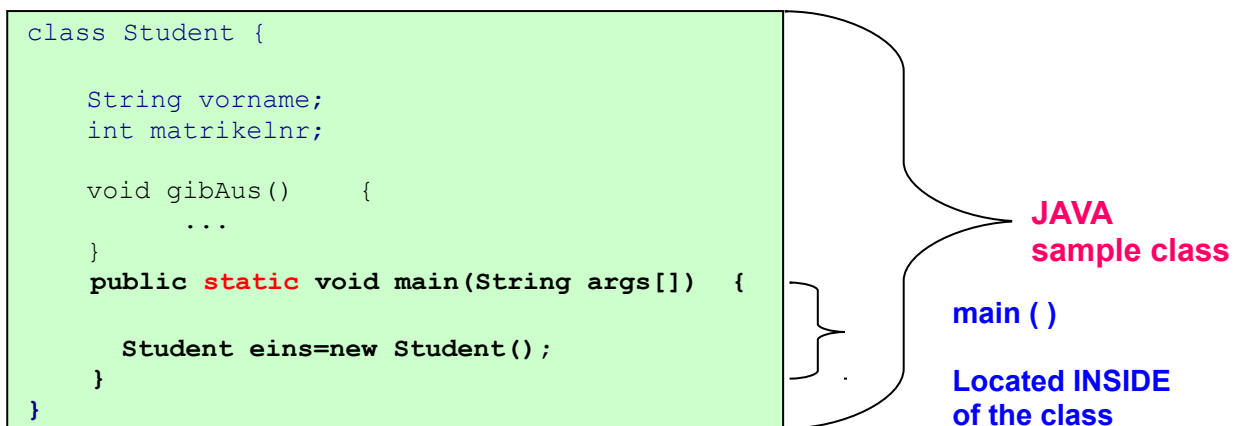
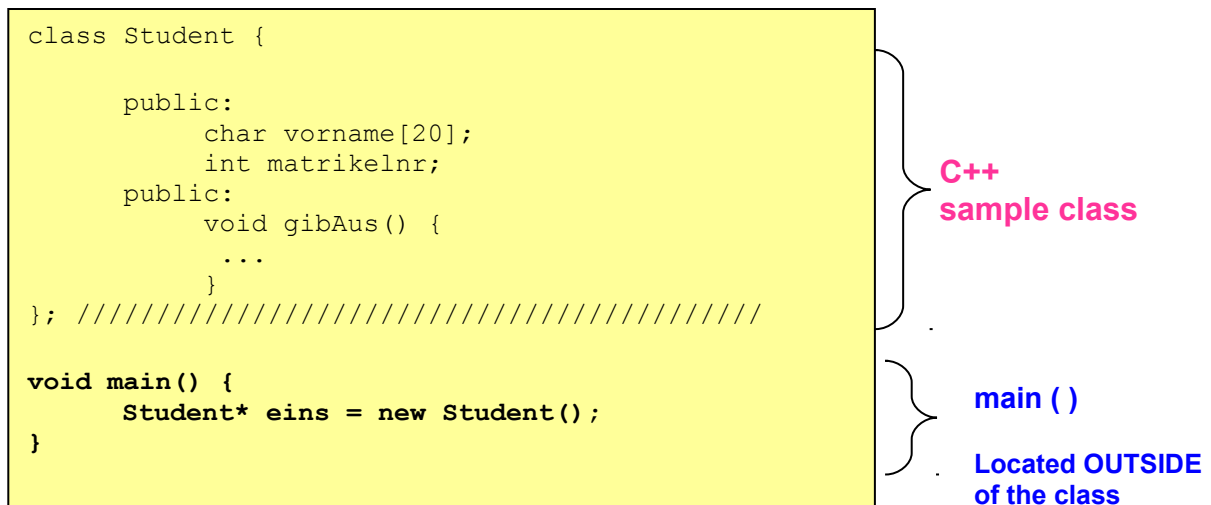
On the developer's computer system the Java sourcecode is written and compiled by the Java compiler into bytecode. The developer's computer system thus needs a Java Software Development Kit (**SDK**) which includes a Runtime Environment to run and test Java programs.

On the client side a **JRE** (Java Runtime Environment) is needed to bring the bytecode to execution. This may happen on any computer system, which has a Java interpreter installed (Windows, Mac, Unix ...). The JRE contains an interpreter and a complete API system library that contains all the standard classes - but no Java compiler.



Structure of a Java program

The strict object-orientation of Java has to be respected when creating source code. Since a class forms the uppermost hierarchical level, all other components (instructions, methods and definitions) must be defined inside a class. Very few instructions are allowed outside a class (e.g. `import`, `package`). The exact definition and usage of Java classes will be explained in a later chapter.



main ()

Just as in C the execution of a Java program starts with a function called `main()`. The main method must be located in a class and follows a specific syntax. As it is directly executable by the JRE without any existing object it must be declared as `static`. It is `public` and `void` (does not return a value) because it is executed by the JRE and not by the operating system.

`Main ()` may receive parameters from the command line and therefore it must declare a string array for handling the arguments. Although this String array actually need not be used in main, it must always be specified. For more detailed explanation about arrays and their attributes / methods see later chapter on Java arrays.

```
// Example program: main with parameter transfer
// from the command line

class Test {

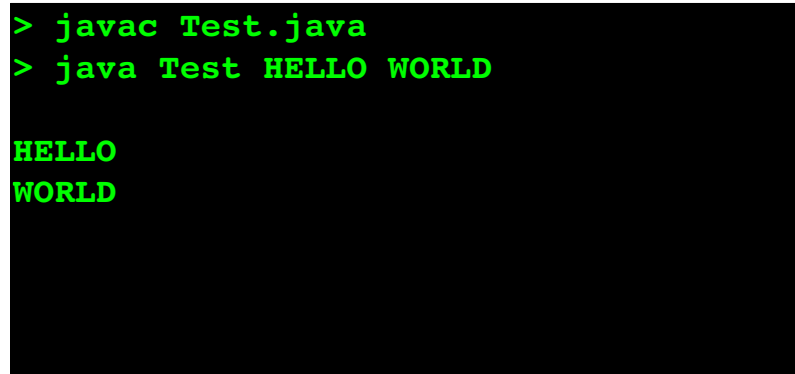
    public static void main (String [] args) {

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

            System.out.println (args [i]);

    }

}
```



```
> javac Test.java
> java Test HELLO WORLD

HELLO
WORLD
```