2. Introduction to Java

Java

- Conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems Inc. in 1991
- Initially called “Oak”.
- Renamed as **Java** in 1995.

Motivation

- Need for a platform-independent, simple language that could be used to create s/w to be embedded in various consumer electronic devices such as microwave ovens, cell phones, washing machines etc.
- Second thrust to the popularity of Java came from the World Wide Web.

Applications and Applets

- Java can be used to create 2 types of programs:
  - **Applications** – program that runs on a computer under the operating system of that computer.
  - **Applets** – program designed to be transmitted over the internet and executed by a Java-compatible Web browser.

Features of Java

- Simple
- Secure
- Portable
- Object-Oriented
- Robust
- Multithreaded
- Architecture neutral
- Distributed

Features of Java

- Simple
  - Designed to be simple.
  - No pointers
- Secure
  - Runs inside JVM – No virus threat
  - Applets are not allowed to access files

Features of Java

- Portable
  - Runs on any hardware and software systems with JVM
  - Suitable for Internet
- Object-Oriented
  - Purely Object-Oriented

Features of Java

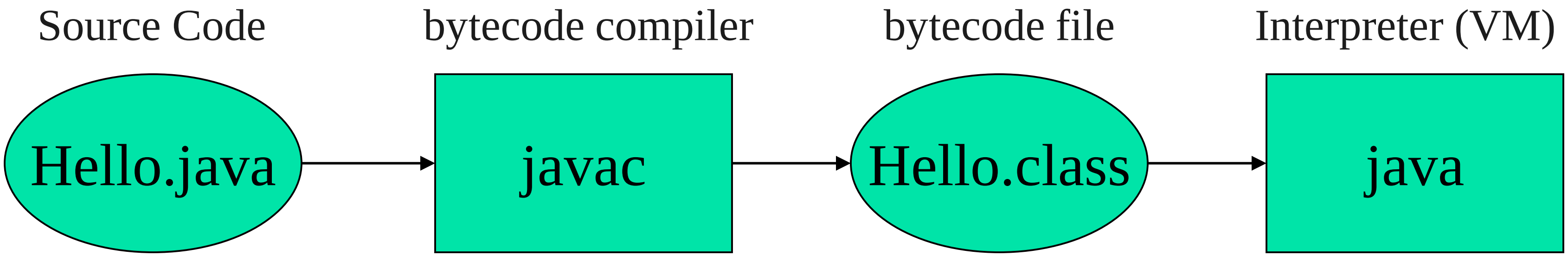
- Robust
  - Compulsory Exception handling
  - Automatic garbage collection
- Multithreaded
  - Doing many things simultaneously
  - Multi threading is built into the language itself

Features of Java

- Architecture Neutral
  - Insensitive to changes in the hardware platform, software upgrades etc.
  - “Write once; run anywhere, any time, forever”
- Distributed
  - Supports network protocols
  - Using a URL is like accessing a file.

Platform independence: What is it? How is it achieved?

- Java is an interpreted language
  - Java source code is translated into Java bytecodes by Java **compiler**
  - This forms the instructions to a Java Virtual Machine (JVM)
  - The JVM interprets the bytecodes and as a result, executes real machine instructions

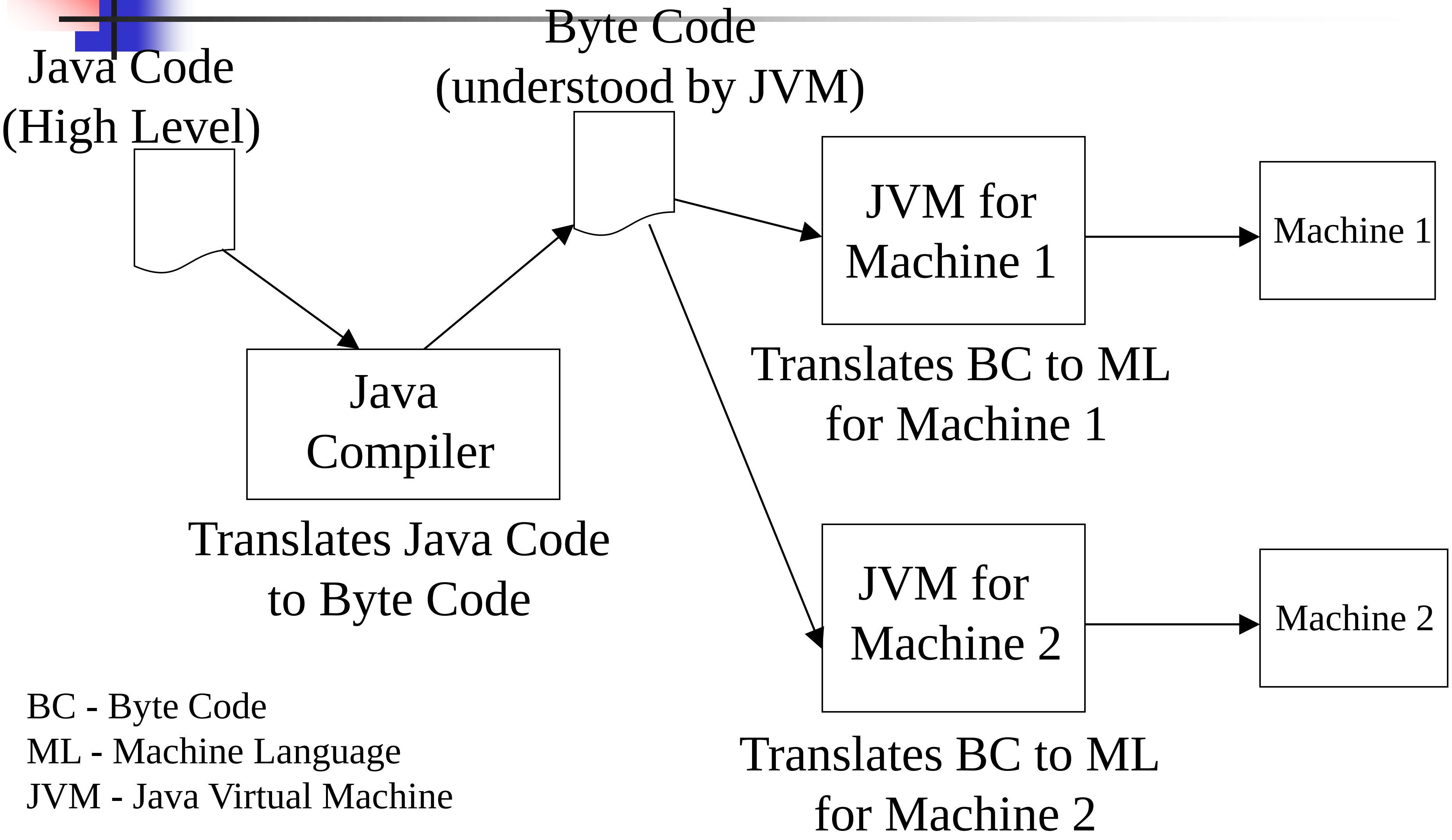


Source Code — bytecode compiler — bytecode file — Interpreter (VM)

Hello.java → javac → Hello.class → java

Platform independence:

- Java bytecodes are machine independent
  - This means that you can
    - compile Java source code on one platform
    - Transfer the bytecodes to another platform and
    - Execute them using a Java VM that has been ported for that platform
  - You can also send Java bytecodes around the Internet as data.



Java Code
(High Level)

Java
Compiler

Translates Java Code
to Byte Code

Byte Code
(understood by JVM)

JVM for
Machine 1

Translates BC to ML
for Machine 1

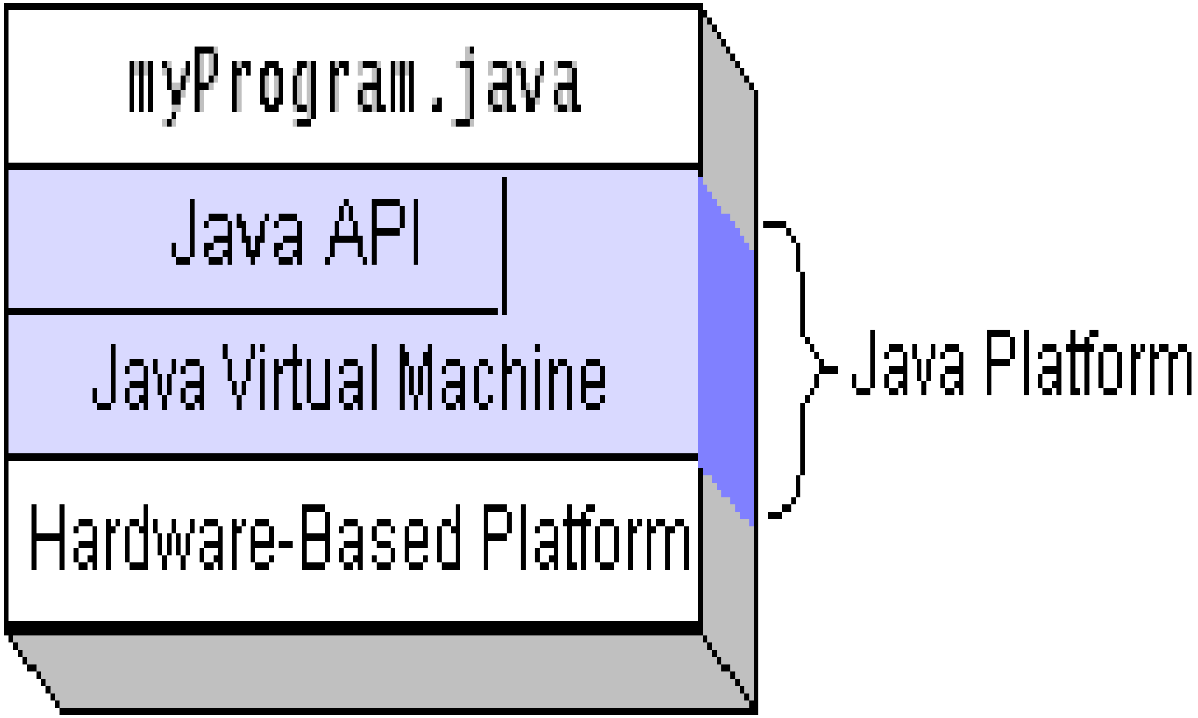
Machine 1

JVM for
Machine 2

Translates BC to ML
for Machine 2

Machine 2

BC - Byte Code
ML - Machine Language
JVM - Java Virtual Machine



myProgram.java

Java API

Java Virtual Machine

Hardware-Based Platform

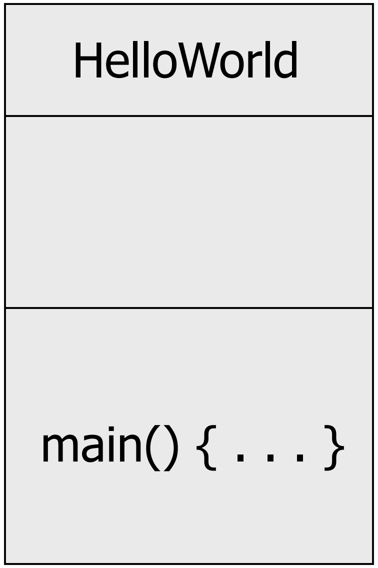
Java Platform

JVM

Java achieves the concept of platform-independence by *inventing* an abstract computer of its own and running on that. This *virtual machine*, as it is called runs a special set of instructions, called *bytecodes* that are simply a stream of formatted bytes, each of which has a precise specification of exactly what each bytecode does to this virtual machine.

The JVM is defined as **"An abstract machine that is implemented by emulating it in software on a real machine".**

A Simple Java Program



HelloWorld

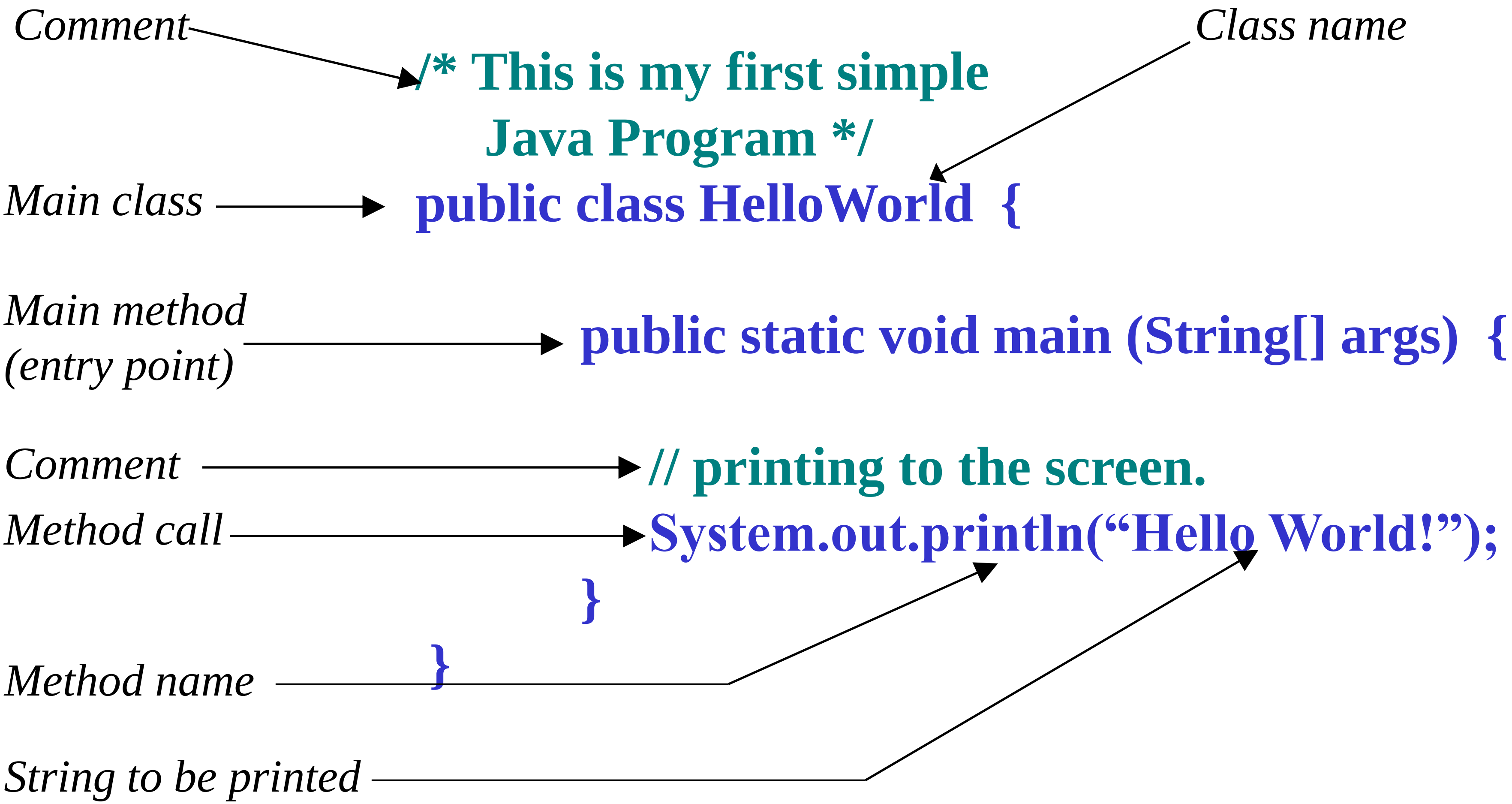
main() { . . . }

```
/* This is my first simple
    Java Program */
public class HelloWorld  {

        public static void main (String[] args)  {

            // printing to the screen.
            System.out.println(“Hello World!”);
        }
 }
```

A Simple Java Program



```
/* This is my first simple
     Java Program */
public class HelloWorld  {

          public static void main (String[] args)  {

                  // printing to the screen.
                  System.out.println(“Hello World!”);
          }
   }
```

Comment

Class name

Main class

Main method (entry point)

Comment

Method call

Method name

String to be printed

Coding Conventions

- Naming Conventions
  - Classes, Objects, Variables, Methods
- Indentations
- Parenthesis

Steps to execute Java Program

- Edit this Java source code and save the file as “HelloWorld.java”
- Compile using Java Compiler.
  - ***javac* HelloWorld.java** – HelloWorld.class
- Interpret and run the program using Java Interpreter.
  - ***java* HelloWorld**

Basic Rules

- The name of the file must always be the name of the “public class”
- Java is case sensitive
- We can have only one public class in a file

Basic Rule

- Every “stand alone” Java program must have a *public static void main()* method defined
  - It is the starting point of the program.
  - *public* ⟶ anyone can call this method
  - *static* ⟶ static method, so we don’t need an instance to call this method.
  - *void* ⟶ the return type is void
  - *main* ⟶ Name of the startup method.

References

- Herbert Schildt, Java 2: The Complete Reference, 7th Edition., McGraw-Hill Publishing Company Inc
- "The Java Tutorial", found online at https://docs.oracle.com/javase/tutorial/tutorialLearningPaths.html