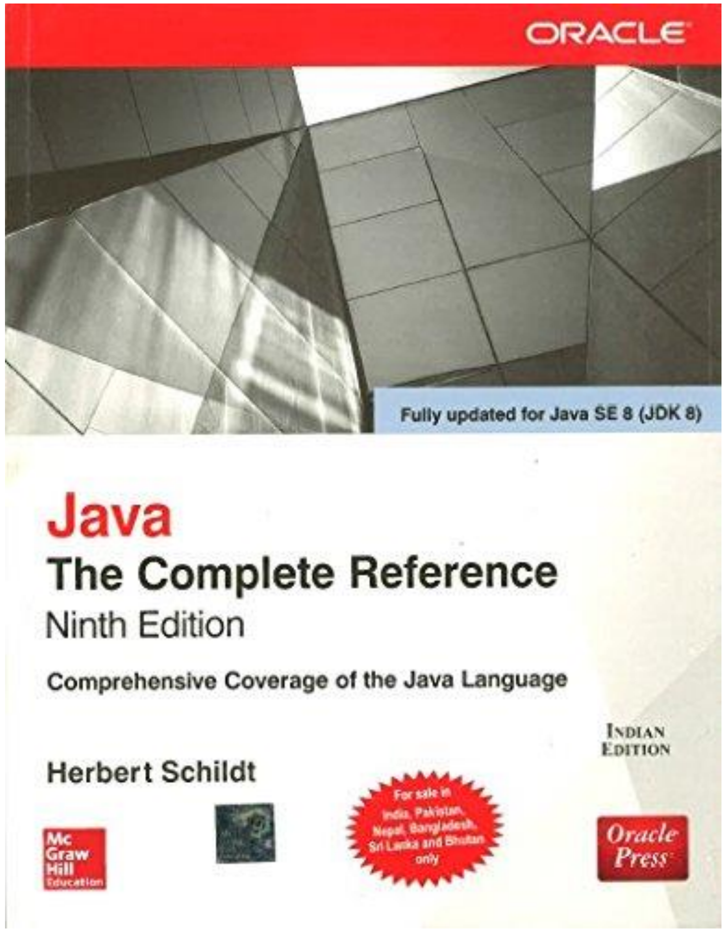
## Orientation

# Object Oriented Programming II (JAVA)

**Md. Mamun Hossain**
B.Sc. (Engg.) & M.Sc. (Thesis) in CSE , SUST
Assistant Professor, Dept. of CSE, BAUST

It is not enough just to write code that works. It is as important-perhaps more important to write code well; not merely code that works, but code that is legible, maintainable, reusable, fast, and efficient.

# References


ORACLE
Fully updated for Java SE 8 (JDK 8)
Java
The Complete Reference
Ninth Edition
Comprehensive Coverage of the Java Language
Indian Edition
Herbert Schildt
For sale in India, Pakistan, Nepal, Bangladesh, Sri Lanka and Bhutan only
Mc Graw Hill Education
Oracle Press

**Basic Text:**

- Java - The Complete Reference: Ninth Edition - By Herbert Schildt.

**Reference Books:**

- Java How To Program (late objects) by Paul Deitel, Harvey Deitel
- Head First Java, 2nd Edition by Kathy Sierra, Bert Bates

**Reference Website**:

- www.javatpoint.com, https://www.tutorialspoint.com,
- https://docs.oracle.com/javase/tutorial/java/
- YouTube channel : PyAcademy2020

**Md. Mamun Hossain**
B.Sc. (Engg.) & M.Sc. (Thesis) in CSE , SUST
Assistant Professor, Dept. of CSE, BAUST

# Course Objectives

- ✓ The Java Programming Language: its syntax, idioms, patterns and styles.
- ✓ How to write, compile and execute Java programs using all major development tolls.
- ✓ To become comfortable with object oriented programming: Object and Class.
- ✓ How to deal with event driven Graphical User Interface (GUI) programming.
- ✓ To retrieve data from a relational database with Java Database Connectivity (JDBC)
- ✓ How to write Java programs that solve practical, real world, business-oriented problems.
- ✓ Program using java API (Application Programming Interface).
- ✓ Program using Exception Handling, Files and Threads .
- ✓ Program Using swings , JavaFx .

It is not enough just to write code that works. It is as important-perhaps more important to write code well; not merely code that works, but code that is legible, maintainable, reusable, fast, and efficient.

# History of Java

- **Java was developed by a team lead by**

  - **James Gosling** – at Sun Microsystems in 1991
  - Vinod Khosla - Co founder
  - Sun Microsystems was purchased by Oracle in 2010

- **Java was initially called as Oak.**
  - But renamed as java in 1995

- C++ was invented by **Bjarne Stroustrup** at Bell Lab in Murry Hill, New Jersey in 1979
  - Initial name of C++ was C with Classes and renamed to C++ in 1983
- C was invented by **Dennis Ritchie** at Bell Lab in 1970

# What is Java?

- Java is a programming **language** and a **platform**.

➢ **Language**: Java is a high level, robust, secured and object-oriented programming language.

➢ **Platform**: Any hardware or software environment in which a program runs, is known as a platform. Since Java has its own runtime environment (JRE) and API, it is called platform.

## Java Editions

- **J2SE**(Java 2 Standard Edition) - to develop client-side standalone applications or applets.
- **J2ME**(Java 2 Micro Edition ) - to develop applications for mobile devices such as cell phones.
- **J2EE**(Java 2 Enterprise Edition ) - to develop server-side applications such as Java servlets and Java Server Pages.

# Where it (Java) is used?

- According to Sun, 3 billion devices run java. There are many devices where java is currently used. Some of them are as follows:
  - ✓ Desktop Applications such as acrobat reader, media player, antivirus etc.
  - ✓ Web Applications such as  javatpoint.com etc.
  - ✓ Enterprise Applications such as banking applications.
  - ✓ Mobile
  - ✓ Embedded System
  - ✓ Smart Card
  - ✓ Robotics
  - ✓ Games etc.

# Types of Java Applications

- There are mainly 4 type of applications that can be created using java programming:

1) **Standalone Application – J2SE**

It is also known as desktop application or window-based application. An application that we need to install on every machine such as media player, antivirus etc. AWT and Swing are used in java for creating standalone applications.

2) **Web Application – J2SE/J2EE - JavaFx**

An application that runs on the server side and creates dynamic page, is called web application. Currently, servlet, jsp, struts, jsf etc. technologies are used for creating web applications in java.

# Types of Java Applications

3) **Enterprise Application – J2EE**

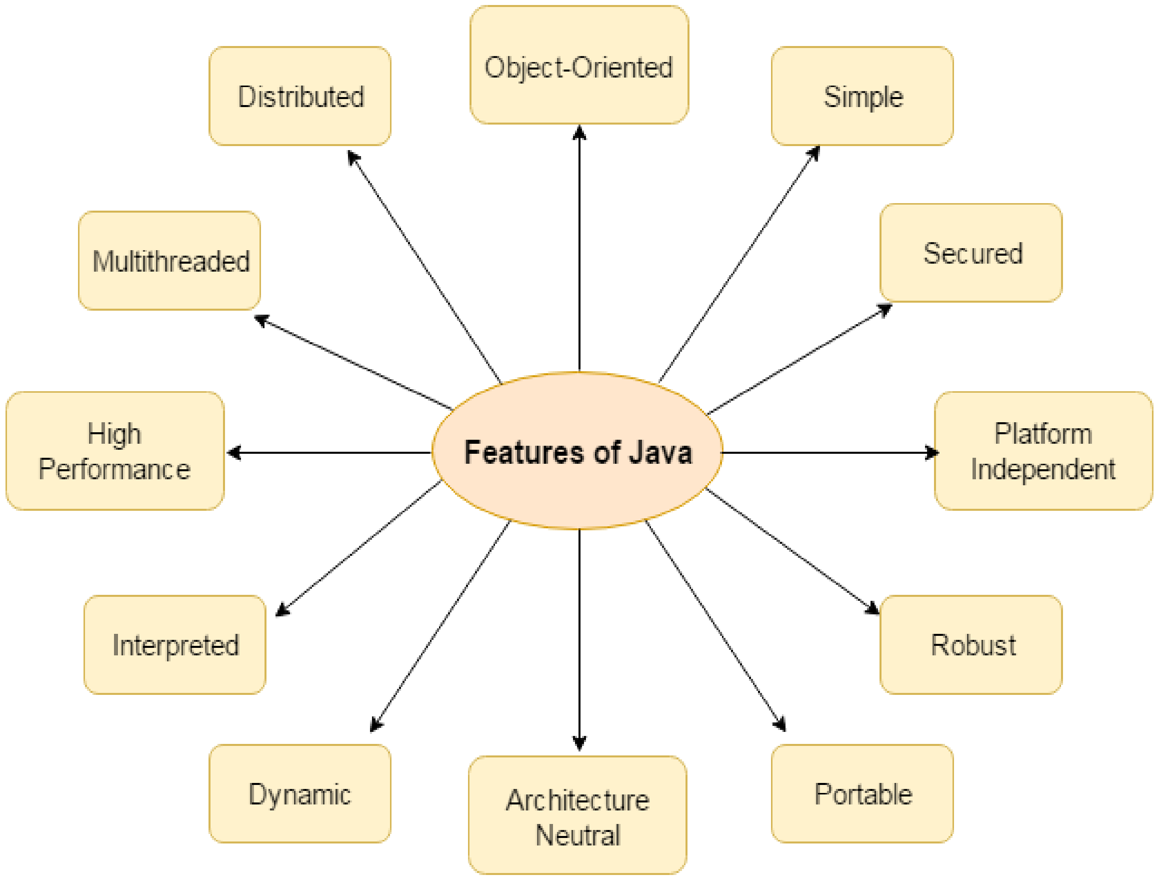
An application that is distributed in nature, such as banking applications etc. It has the advantage of high level security, load balancing and clustering. In java, EJB is used for creating enterprise applications.

4) **Mobile Application -**J2ME

An application that is created for mobile devices. Currently Android and Java ME are used for creating mobile applications.

# Features of java

- There is given many features of java. They are also known as java buzzwords. The Java Features given below are simple and easy to understand.
  - ✓ Simple
  - ✓ Object-Oriented
  - ✓ Portable
  - ✓ Platform independent
  - ✓ Secured
  - ✓ Robust
  - ✓ Architecture neutral
  - ✓ Dynamic
  - ✓ Interpreted
  - ✓ High Performance
  - ✓ Multithreaded
  - ✓ Distributed

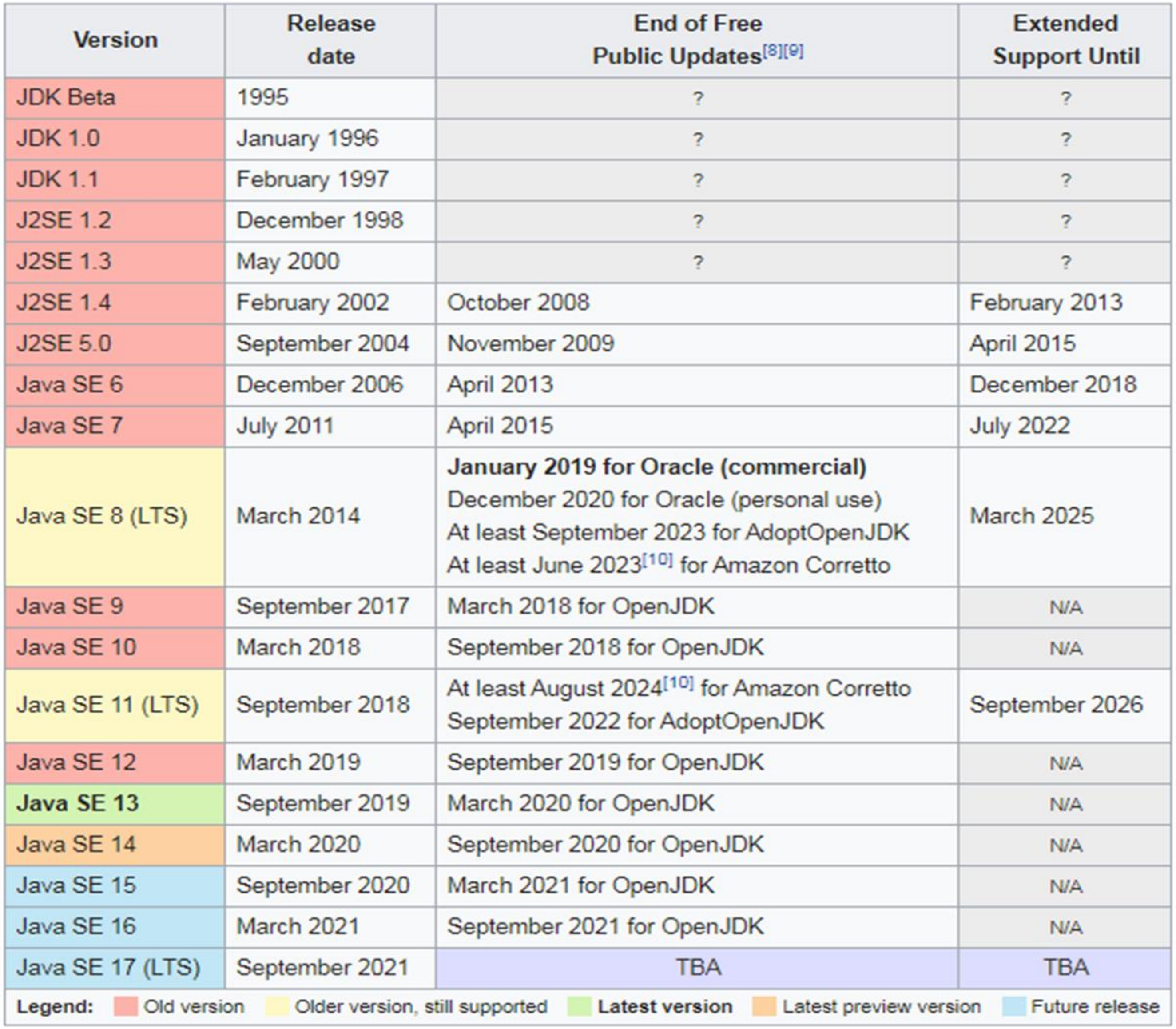


Write Once Run Anywhere (**WORA**).

## Why Java is Important

- Two reasons :
  - Trouble with C/C++ language is that they are not portable and are not platform independent languages.
  - Emergence of World Wide Web, which demanded portable programs
- Portability and security necessitated the invention of Java

# JDK Evolutions



| Version | Release date | End of Free Public Updates[8][9] | Extended Support Until |
|---|---|---|---|
| JDK Beta | 1995 | ? | ? |
| JDK 1.0 | January 1996 | ? | ? |
| JDK 1.1 | February 1997 | ? | ? |
| J2SE 1.2 | December 1998 | ? | ? |
| J2SE 1.3 | May 2000 | ? | ? |
| J2SE 1.4 | February 2002 | October 2008 | February 2013 |
| J2SE 5.0 | September 2004 | November 2009 | April 2015 |
| Java SE 6 | December 2006 | April 2013 | December 2018 |
| Java SE 7 | July 2011 | April 2015 | July 2022 |
| Java SE 8 (LTS) | March 2014 | **January 2019 for Oracle (commercial)**<br>December 2020 for Oracle (personal use)<br>At least September 2023 for AdoptOpenJDK<br>At least June 2023[10] for Amazon Corretto | March 2025 |
| Java SE 9 | September 2017 | March 2018 for OpenJDK | N/A |
| Java SE 10 | March 2018 | September 2018 for OpenJDK | N/A |
| Java SE 11 (LTS) | September 2018 | At least August 2024[10] for Amazon Corretto<br>September 2022 for AdoptOpenJDK | September 2026 |
| Java SE 12 | March 2019 | September 2019 for OpenJDK | N/A |
| **Java SE 13** | September 2019 | March 2020 for OpenJDK | N/A |
| Java SE 14 | March 2020 | September 2020 for OpenJDK | N/A |
| Java SE 15 | September 2020 | March 2021 for OpenJDK | N/A |
| Java SE 16 | March 2021 | September 2021 for OpenJDK | N/A |
| Java SE 17 (LTS) | September 2021 | TBA | TBA |

**Legend:** 🟥 Old version 🟨 Older version, still supported 🟩 **Latest version** 🟧 Latest preview version 🟦 Future release

# Java Environment

- Java includes many development **tools**, **classes** and **methods**
  - Development tools are part of Java Development Kit (JDK) and
  - The classes and methods are part of **Java Standard Library (JSL)**, also known as **Application Programming Interface (API).**
- JDK constitutes of tools like **java compiler**, java interpreter and many.
- **API** includes hundreds of **classes** and **methods** grouped into several **packages** according to their functionality.

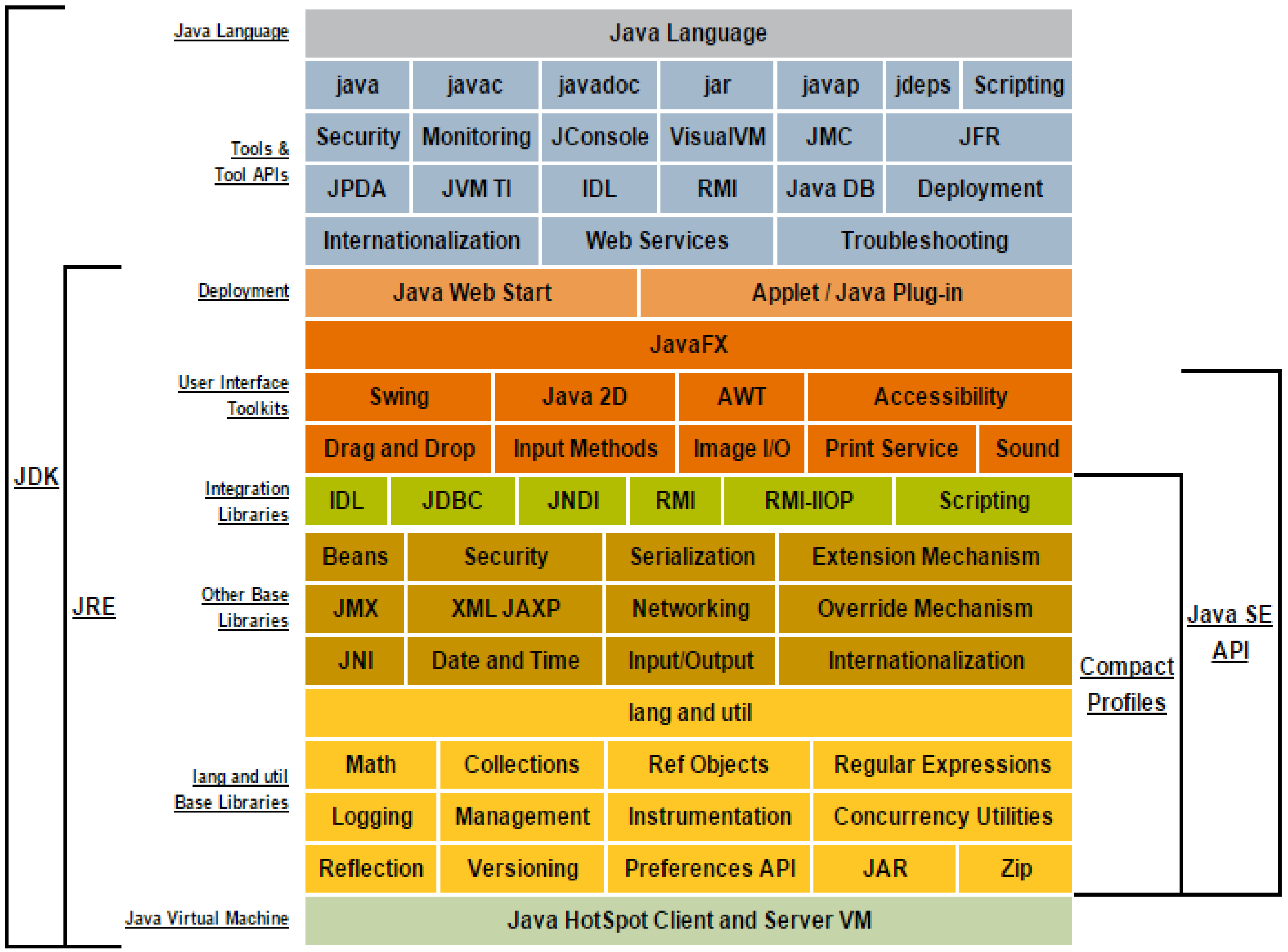
## Java Environment : JDK , JRE & JVM

- **JDK is Java Developer Kit**
  - the JDK is what you need to compile Java source code and contains a JRE, among other things.
  - Java compiler (**javac**), Java Debugger (jdb) etc.
- **JRE is Java Runtime Environment**

  - JRE is what you need to run a Java program and contains a JVM, among other things.

  - API's (classes & methods)
- **JVM is Java Virtual Machine**
  - JVM was designed as an interpreter for bytecode
  - the JVM actually runs Java bytecode (.class file)
  - bytecode is a highly optimized set of instructions designed to be executed by Java Runtime System which is called JVM

# Java Environment : Conceptual Diagram

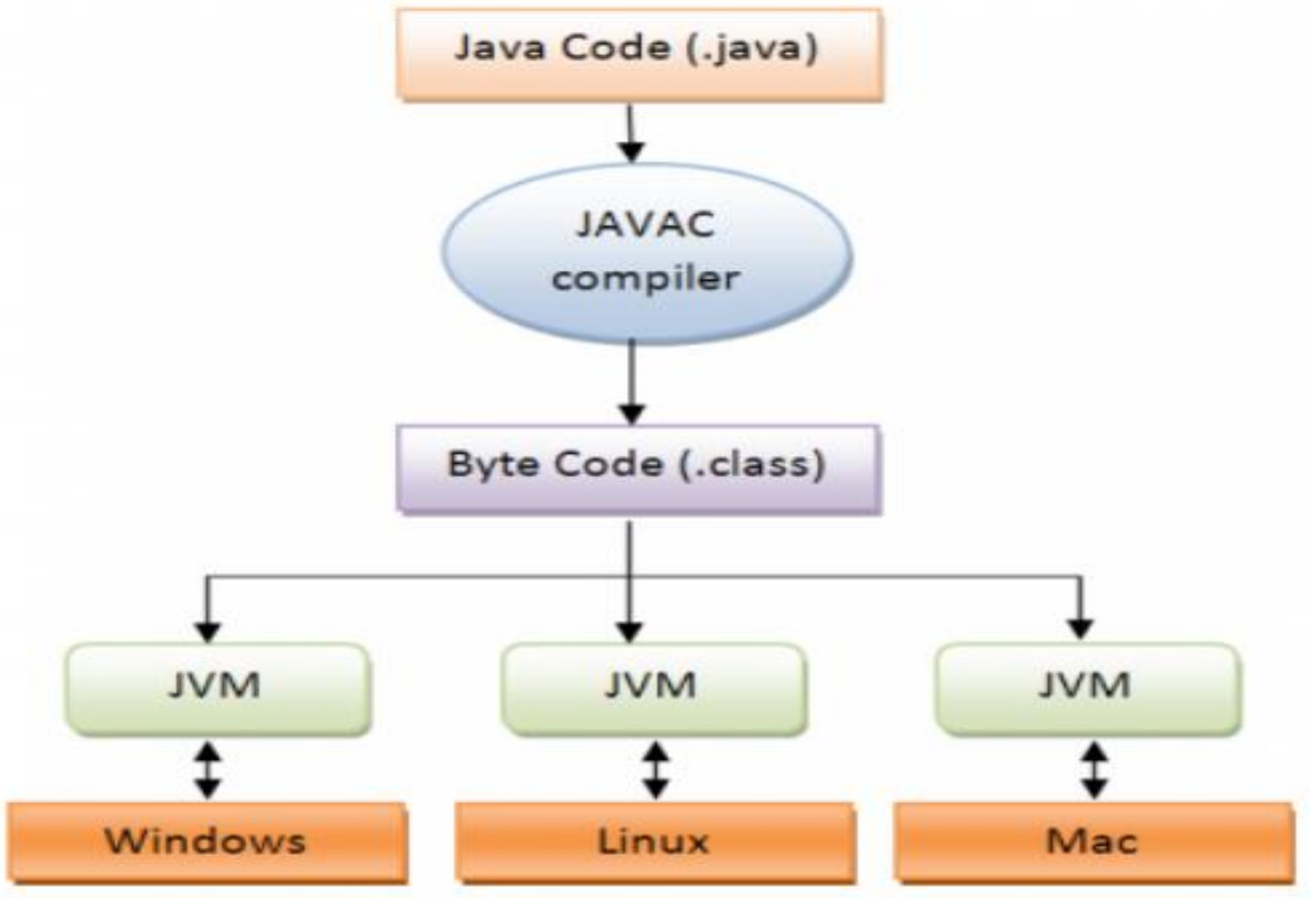


## Byte Code & JVM (Java Virtual Machine)

- Since platform-independence is a defining characteristic of Java, it is important to understand how it is achieved. Programs exist in two forms; source code and object code. Source Code is the textual version of the program that you write using a text editor. The programs printed in a book are shown as source code. The executable form of a program is object code. The computer can execute object code. Typically, object code is specific to a particular CPU. Therefore, it cannot be executed on a different platform. Java removes this feature in a very elegant manner.
- Like all computer languages, a java program begins with its source code. The difference is what happens when a Java program is compiled. Instead of producing executable code, the Java Compiler produces an object file that contains bytecode. Bytecodes are instructions that are not for any specific CPU. Instead, they are designed to be interpreted by a Java Virtual Machine (JVM). The key to Java's platform-independence comes from the fact that the same bytecodes can be executed by any JVM on any platform. As long as there is a JVM implemented for a given environment, it can run any Java program.
- For example, Java programs can execute under Windows 98,Solaris,IRIX, or any other platform for which a JVM can be implemented for that platform. This would then allow any Java program to execute in that new environment.

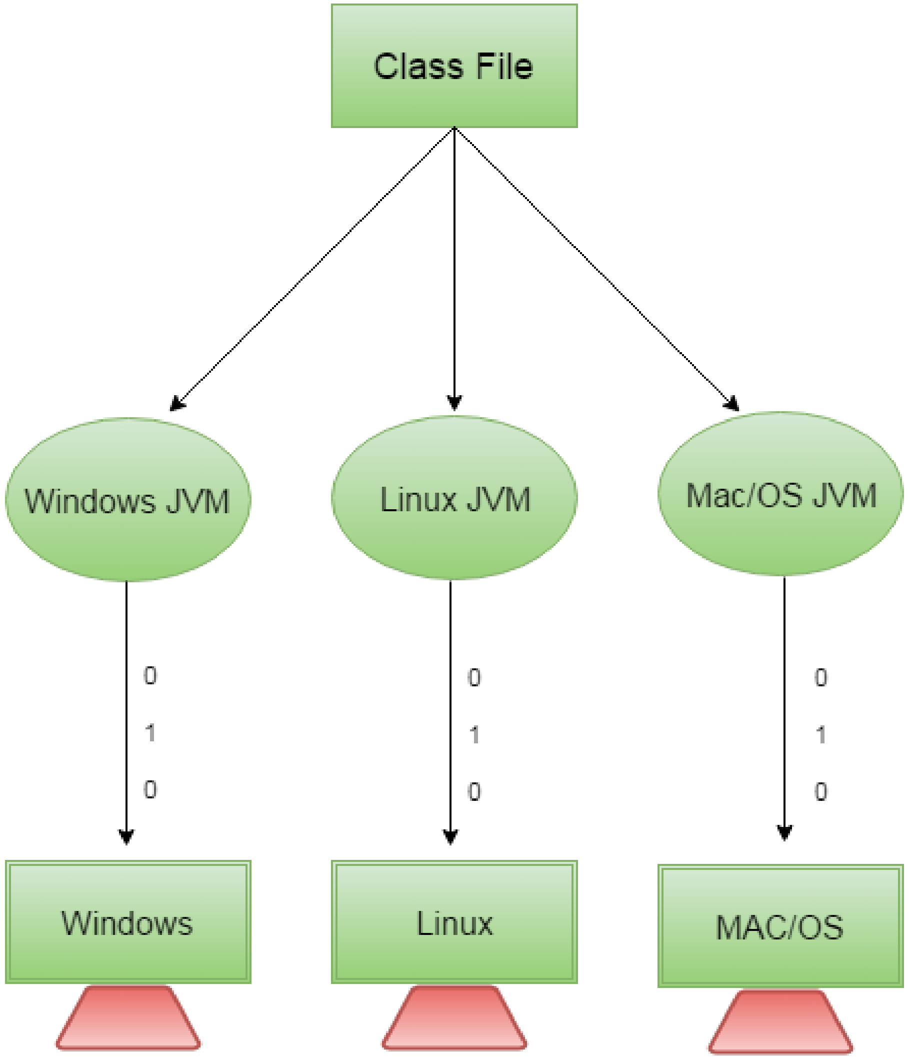
# Java is architecture-neutral

**JAVA Program Execution**

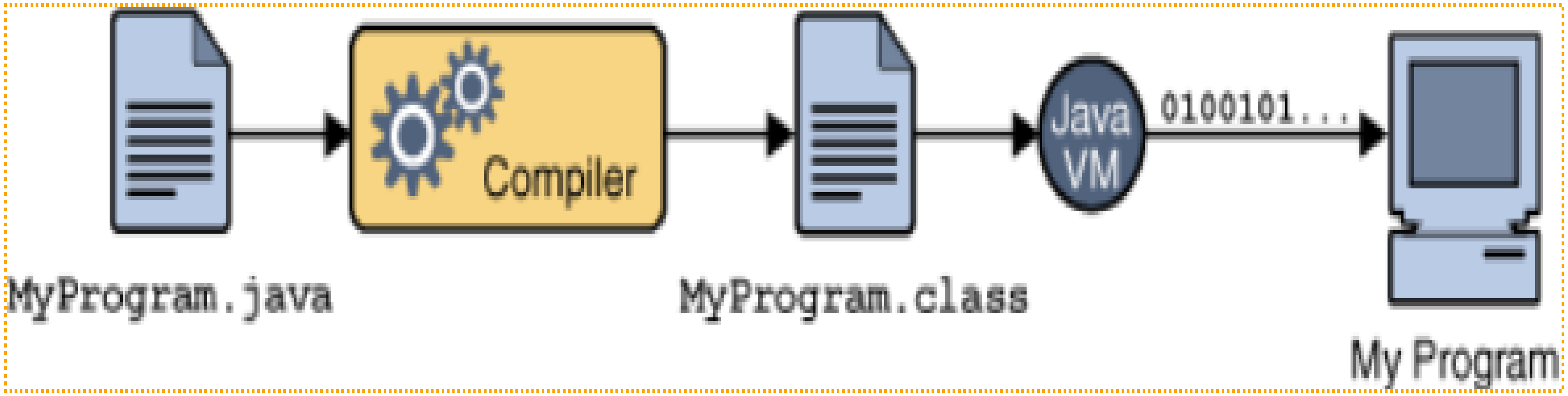


# WORA feature : Platform Independent

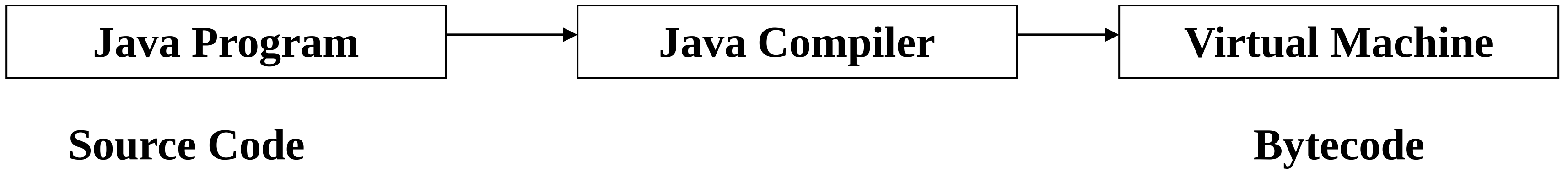
Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere(WORA).



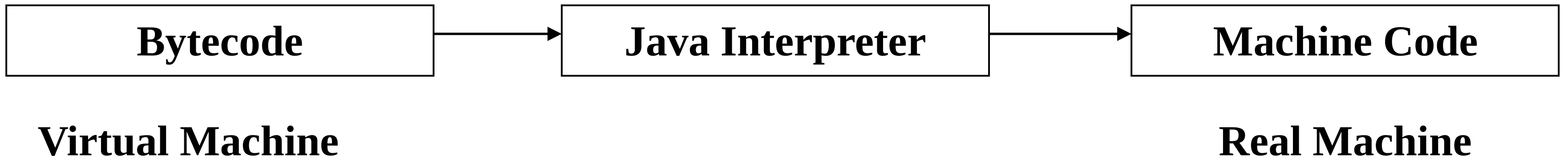
## WORA(Write Once Run Anywhere)



# Process of Compilation

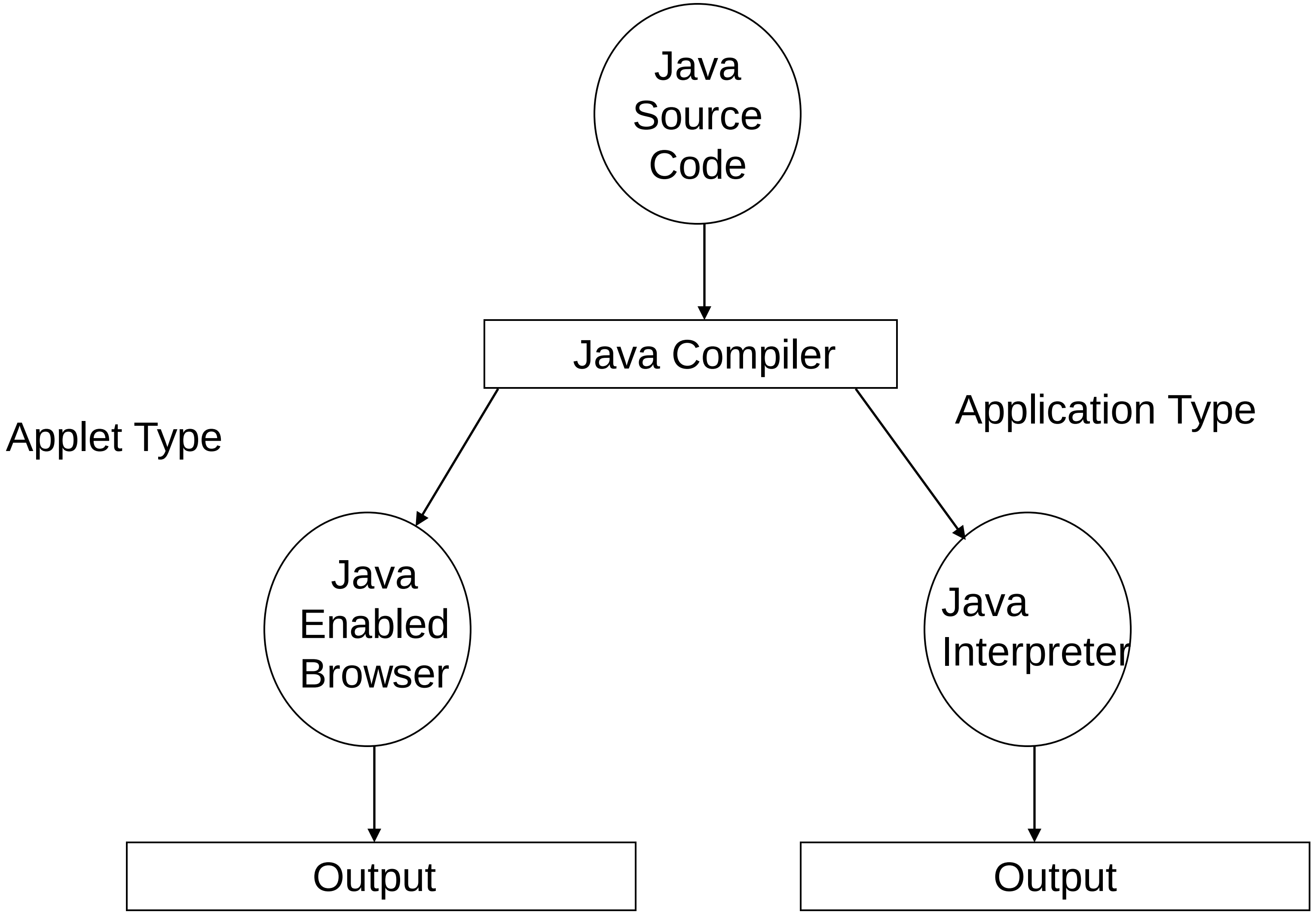


Process of Compilation



Process of Converting bytecode into machine code

# Process of Compilation



## Different Programming Paradigms

- Functional/procedural programming:
  - program is a list of instructions to the computer
  - C, Fortran
- Object-oriented programming
  - program is composed of a collection *objects that communicate with each other*
  - *C++, Java*

# C, C++ , Java

- C→ C++ → Java Java is related to C++
- C++ is direct descendant of C (Superset of C)
- Much of characters of java is inherited from C & C++
  - From C, java drives its syntax
  - From C++, java drives many of its OOP features
- C++'s influence is Java **(but not an enhanced version of C++)**
- Java 's influence is C# (created by Microsoft to support .NET framework)

## How is Java different from C…

- **C Language:**
  - Major difference is that C is a structure oriented language and Java is an object oriented language and has mechanism to define classes and objects.
  - Java does not support an explicit pointer type
  - Java does not have preprocessor, so we cant use #define, #include and #ifdef statements.
  - Java does not include structures, unions and enum data types.
  - Java does not include keywords like goto, sizeof and typedef.
  - Java adds labeled break and continue statements.
  - Java adds many features required for object oriented programming.

## How is Java different from C++…

- **C++ language**

  Features removed in java:

  - Java doesn't support pointers to avoid unauthorized access of memory locations.
  - Java does not include structures, unions and enum data types.
  - Java does not support operator over loading.
  - Preprocessor plays less important role in C++ and so eliminated entirely in java.
  - Java does not perform automatic type conversions that result in loss of precision.

## How is Java different from C++ (Cont.)

- Java does not support global variables. Every method and variable is declared within a class and forms part of that class.
- Java does not allow default arguments.
- Java does not support inheritance of multiple super classes by a sub class (i.e., multiple inheritance). This is accomplished by using ‘interface’ concept.
- It is not possible to declare unsigned integers in java.
- In java objects are passed by reference only. In C++ objects may be passed by value or reference.

# New features added in Java:

- Multithreading, that allows two or more pieces of the same program to execute concurrently.
- C++ has a set of library functions that use a common header file. But java replaces it with its own set of API classes.
- It adds packages and interfaces.
- Java supports automatic garbage collection.
- break and continue statements have been enhanced in java to accept labels as targets.
- The use of unicode characters ensures portability.

## Features that differ

- Though C++ and java supports Boolean data type, C++ takes any nonzero value as true and zero as false. True and false in java are predefined literals that are values for a boolean expression.
- Java has replaced the destructor function with a finalize() function.
- C++ supports exception handling that is similar to java's. However, in C++ there is no requirement that a thrown exception be caught.

# C++ vs. Java

| Comparison Index | C++ | Java |
|---|---|---|
| Platform-independent | C++ is platform-dependent. | Java is platform-independent. |
| Mainly used for | C++ is mainly used for system programming. | Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications. |
| Goto | C++ supports goto statement. | Java doesn't support goto statement. |
| Multiple inheritance | C++ supports multiple inheritance. | Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java. |

# C++ vs. Java

| Comparison Index | C++ | Java |
|---|---|---|
| Operator Overloading | C++ supports operator overloading. | Java doesn't support operator overloading. |
| Pointers | C++ supports pointers. You can write pointer program in C++. | Java supports pointer internally. But you can't write the pointer program in java. It means java has restricted pointer support in java. |
| Compiler and Interpreter | C++ uses compiler only. | Java uses compiler and interpreter both. |
| Call by Value and Call by reference | C++ supports both call by value and call by reference. | Java supports call by value only. There is no call by reference in java. |
| Structure and Union | C++ supports structures and unions. | Java doesn't support structures and unions. |

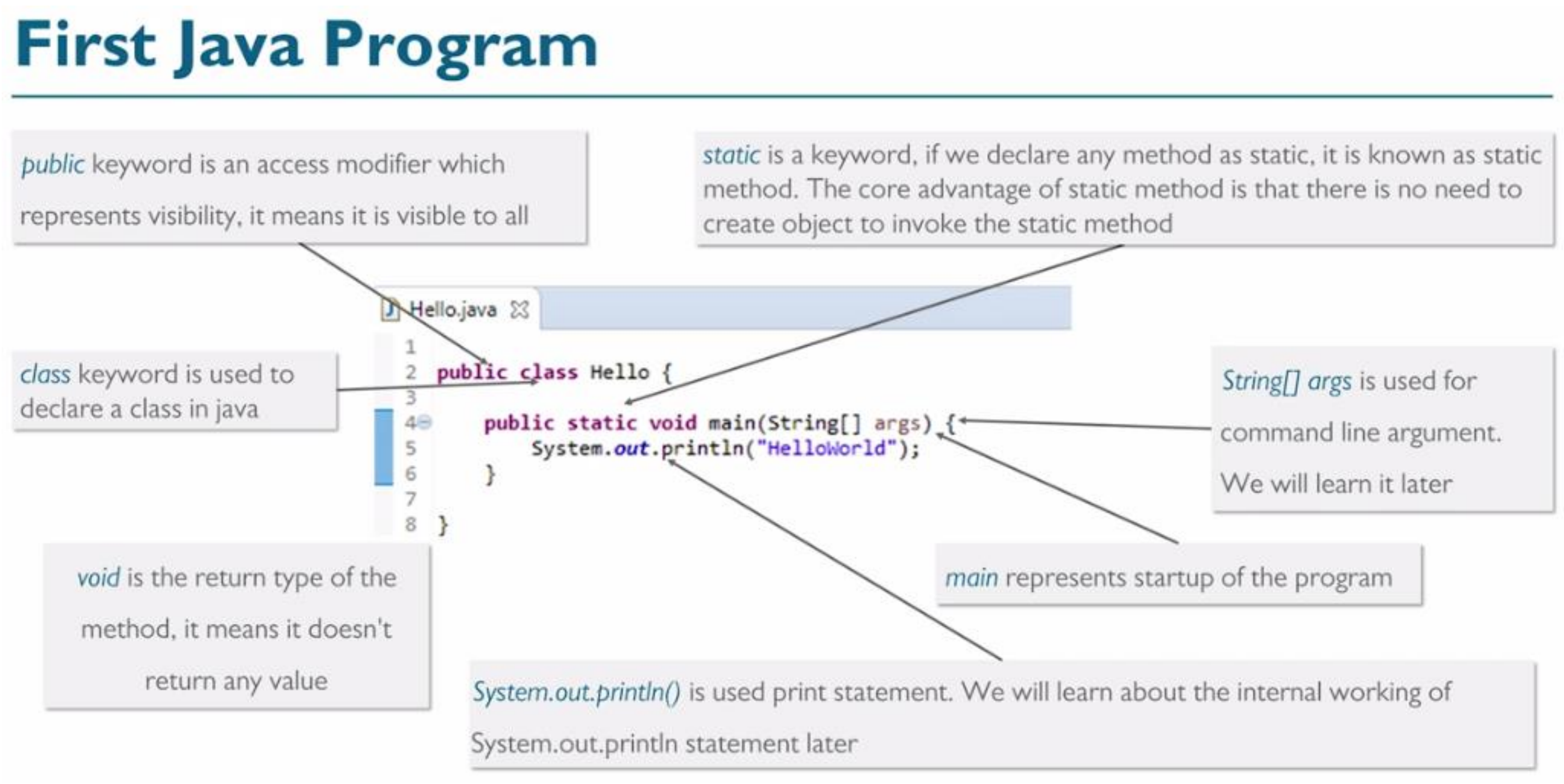
# C++ vs. Java

| Comparison Index | C++ | Java |
|---|---|---|
| Thread Support | C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support. | Java has built-in thread support. |
| Documentation comment | C++ doesn't support documentation comment. | Java supports documentation comment (/** ... */) to create documentation for java source code. |
| Virtual Keyword | C++ supports virtual keyword so that we can decide whether or not override a function. | Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default. |

# C++ vs. Java

| Comparison Index | C++ | Java |
|---|---|---|
| unsigned right shift >>> | C++ doesn't support >>> operator. | Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator. |
| Inheritance Tree | C++ creates a new inheritance tree always. | Java uses single inheritance tree always because all classes are the child of Object class in java. Object class is the root of inheritance tree in java. |

# Hello World Program



## First Java Program

*public* keyword is an access modifier which represents visibility, it means it is visible to all

*static* is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method

*class* keyword is used to declare a class in java

*String[] args* is used for command line argument. We will learn it later

Hello.java

```java

public class Hello {

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

}
```

*void* is the return type of the method, it means it doesn't return any value

*main* represents startup of the program

*System.out.println()* is used print statement. We will learn about the internal working of System.out.println statement later