INTRODUCTION

Outcome of the Lecture

Upon completion of this lecture you will be able to understand

- ✓ Fundamentals and Characteristics of Java Language
- ✓ Basic Terminology – JVM, JRE, API, JDK, IDE, bytecode
- ✓ Writing, Compiling and Executing Java Program

Outline of the Presentation

- ✓ Overview of Java
- ✓ History of Java
- ✓ Java Standards
- ✓ Editions of Java
- ✓ JDK and IDE
- ✓ Characteristics of Java
- ✓ Sample Java Program
- ✓ Creating, Compiling and Executing Java Program
- ✓ Java Runtime Environment (JRE) and Java Virtual Machine (JVM)

Overview of Java

- A general-purpose Object-Oriented language
- Developed by Sun Microsystems, later acquired by Oracle Corporation
- Java can be used to develop
  - ✓ Stand alone applications
  - ✓ Web applications.
  - ✓ applications for hand-held devices such as cell phones

History of Java

- Developed by a team led by James Gosling at Sun Microsystems in 1991 for use in embedded chips .
- The primary motivation was the need for a platform-independent (that is, architecture-neutral) language that could be used to create software to be embedded in various consumer electronic devices, such as microwave ovens and remote controls.
- Originally called Oak.
- In 1995 redesigned for developing Internet applications and renamed as Java
- HotJava - The first Java-enabled Web browser (1995)

Java Standards

Computer languages have strict rules of usage. Java standards are defined by

- The Java language specification and
- Java API

The *Java language specification* is a technical definition of the language that includes the syntax and semantics of the Java programming language.

The *Application Program Interface (API)* contains predefined classes and interfaces for developing Java programs(or Java library)

Java language specification is stable, but the API is still expanding

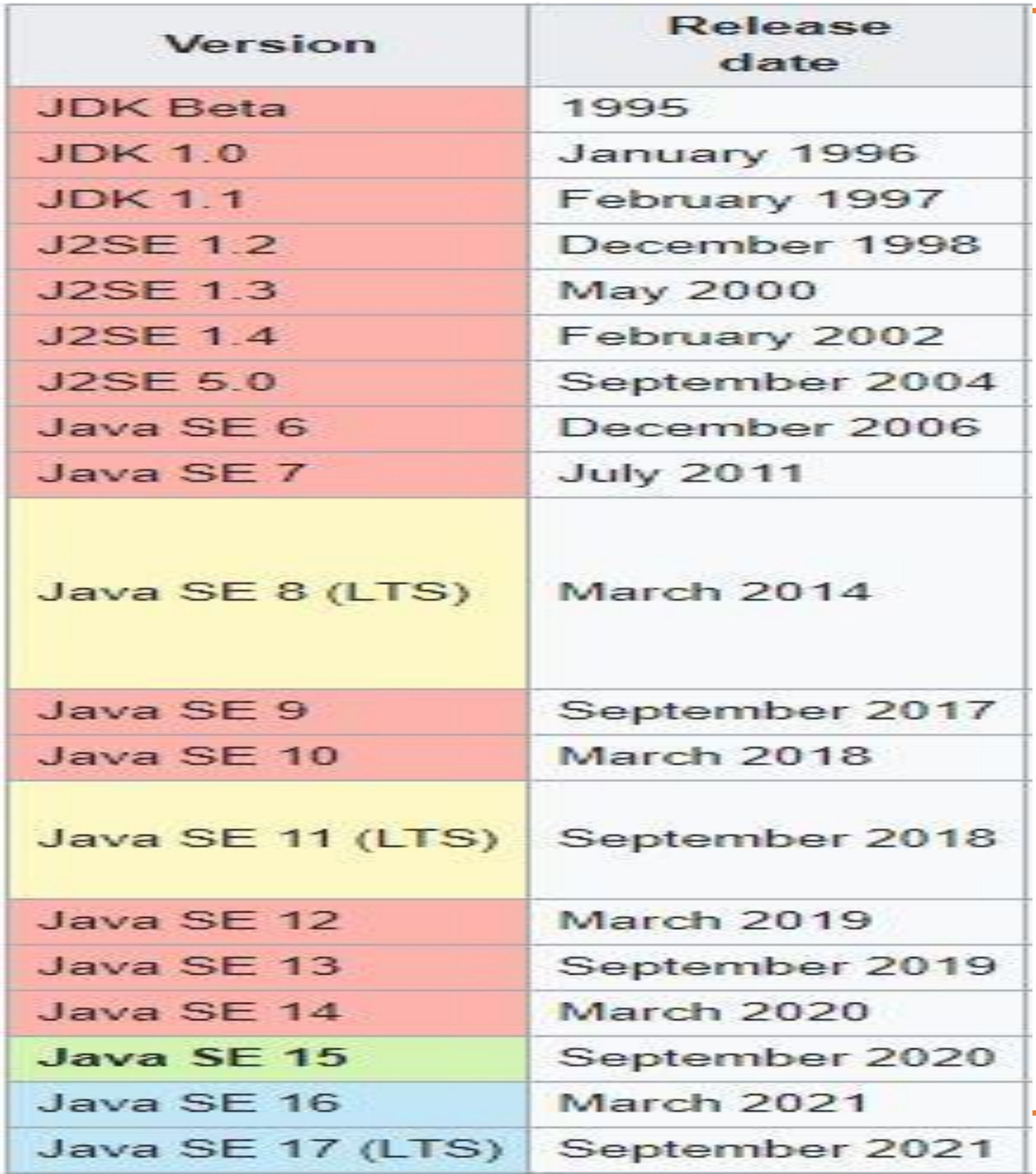
Editions of Java

- ✓ Java Standard Edition (SE)
  used to develop client-side standalone applications or applets.
- ✓ Java Enterprise Edition (EE)
  used to develop server-side applications such as Java servlets and Java ServerPages.
- ✓ Java Micro Edition (ME).
  used to develop applications for mobile devices such as cell phones.

We will be using Java SE

Various versions of Java over the years

CODERINDEED



Version	Release date
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
J2SE 5.0	September 2004
Java SE 6	December 2006
Java SE 7	July 2011
Java SE 8 (LTS)	March 2014
Java SE 9	September 2017
Java SE 10	March 2018
Java SE 11 (LTS)	September 2018
Java SE 12	March 2019
Java SE 13	September 2019
Java SE 14	March 2020
Java SE 15	September 2020
Java SE 16	March 2021
Java SE 17 (LTS)	September 2021

Java
ORACLE

Integrated Development Environment (IDE)

- ✓ Besides JDK, a Java development tool—software that provides an integrated development environment (IDE) for rapidly developing Java programs can be used.

- ✓ Editing, compiling, building, debugging, and online help are integrated in one graphical user interface.

  - Borland Jbuilder
  - Microsoft Visual J++
  - Net Beans
  - Eclipse
  - BlueJ

Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Secure
- Java Is Robust
- Java Is Interpreted
- Java Is Distributed
- Java Is Architecture-Neutral
- Java Is Portable
- Java's Performance
- Java Is Multithreaded
- Java Is Dynamic

Java Is Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun, Java language is a simple programming language because:

- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

Characteristics of Java

Java Is Object-Oriented

Java is inherently object-oriented. Object-oriented programming provides great flexibility, modularity, clarity, and reusability through encapsulation, inheritance, and polymorphism

Java Is Secure

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

- No explicit pointer
- Java Programs run inside a virtual machine sandbox

Java Is Robust

- Robust simply means strong. Java is robust because:
- It uses strong memory management.
- There is a lack of pointers that avoids security problems.
- There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

Java Is Distributed

Java is distributed because it facilitates users to create distributed applications in Java. RMI's are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

Java Is Interpreted(While execution)

You need an interpreter to run Java programs. The programs are compiled into the Java Virtual Machine code called bytecode. The bytecode is machine-independent and can run on any machine that has a Java interpreter, which is part of the Java Virtual Machine (JVM).

Java Is Architecture-Neutral

Write once, run anywhere. With a Java Virtual Machine (JVM), you can write one program that will run on any platform.

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.

•In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

Java Is Portable

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation

Characteristics of Java

Java’s Performance

- Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code.

Java provides Multithreading

- A thread allows multiple activities within a single process. We can write Java programs that deal with many tasks at once by defining multiple threads.
- The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

Java is Dynamic

- Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

JDK,JRE and JVM

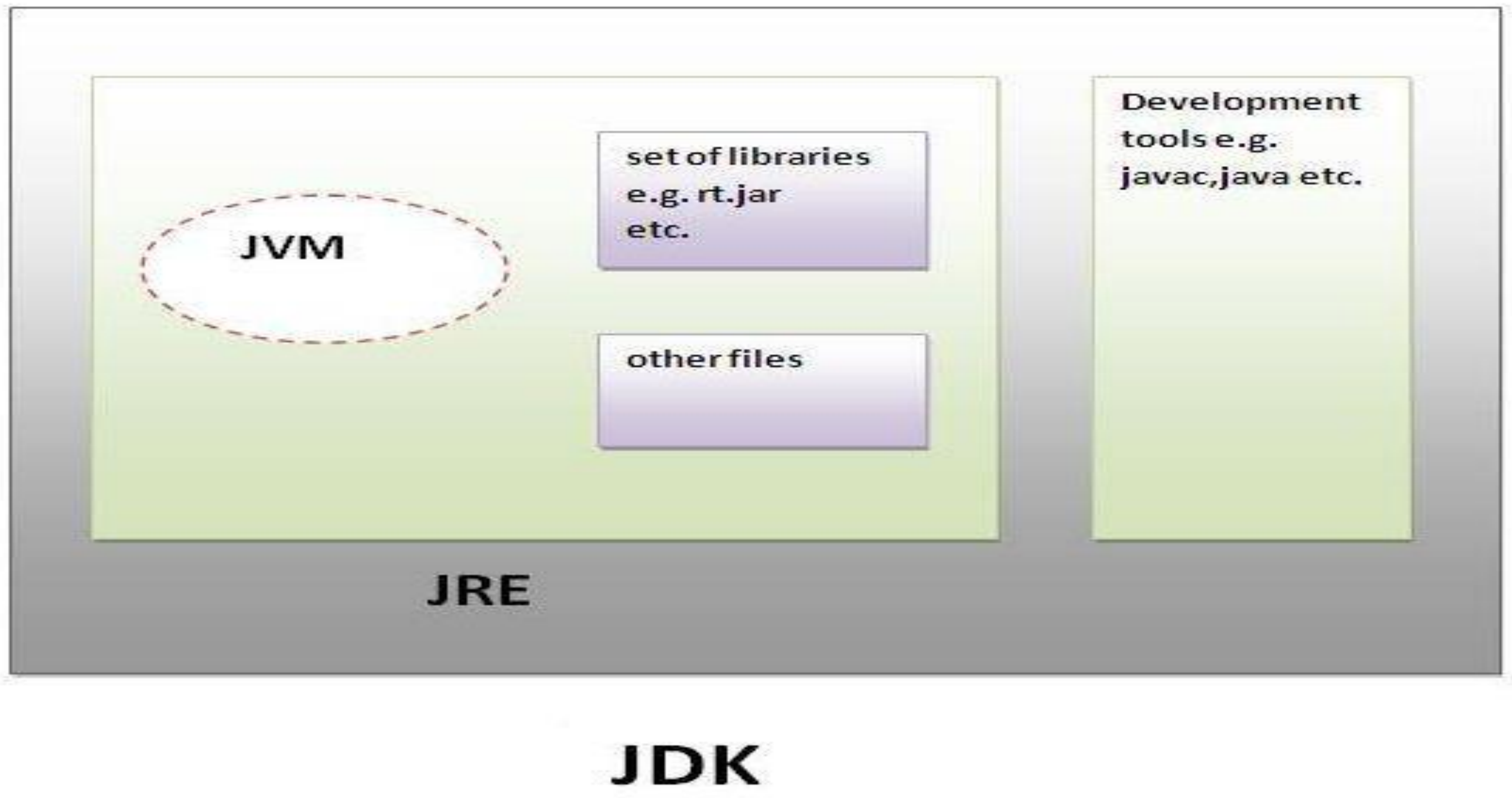
- **JDK – Java Development Kit** (in short JDK) is Kit which provides the environment to **develop and execute(run)** the Java program. JDK is a kit(or package) which includes two things
  - Development Tools(to provide an environment to develop your java programs)
  - JRE (to execute your java program).

  Note : JDK is only used by Java Developers.

- **JRE – Java Runtime Environment** (to say JRE) is an installation package which provides environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by them who only wants to run the Java Programs i.e. end users of your system.

- **JVM – Java Virtual machine**(JVM) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for **executing the java program line by line** hence it is also known as interpreter.

Understanding JDK, JRE and JVM


JVM
set of libraries e.g. rt.jar etc.
other files
Development tools e.g. javac,java etc.
JRE
JDK

Various components of JDK & JRE

JDK

- JDK is an acronym for *Java Development Kit.*
- It physically exists. It contains JRE and development tools.
- The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets.
- It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.

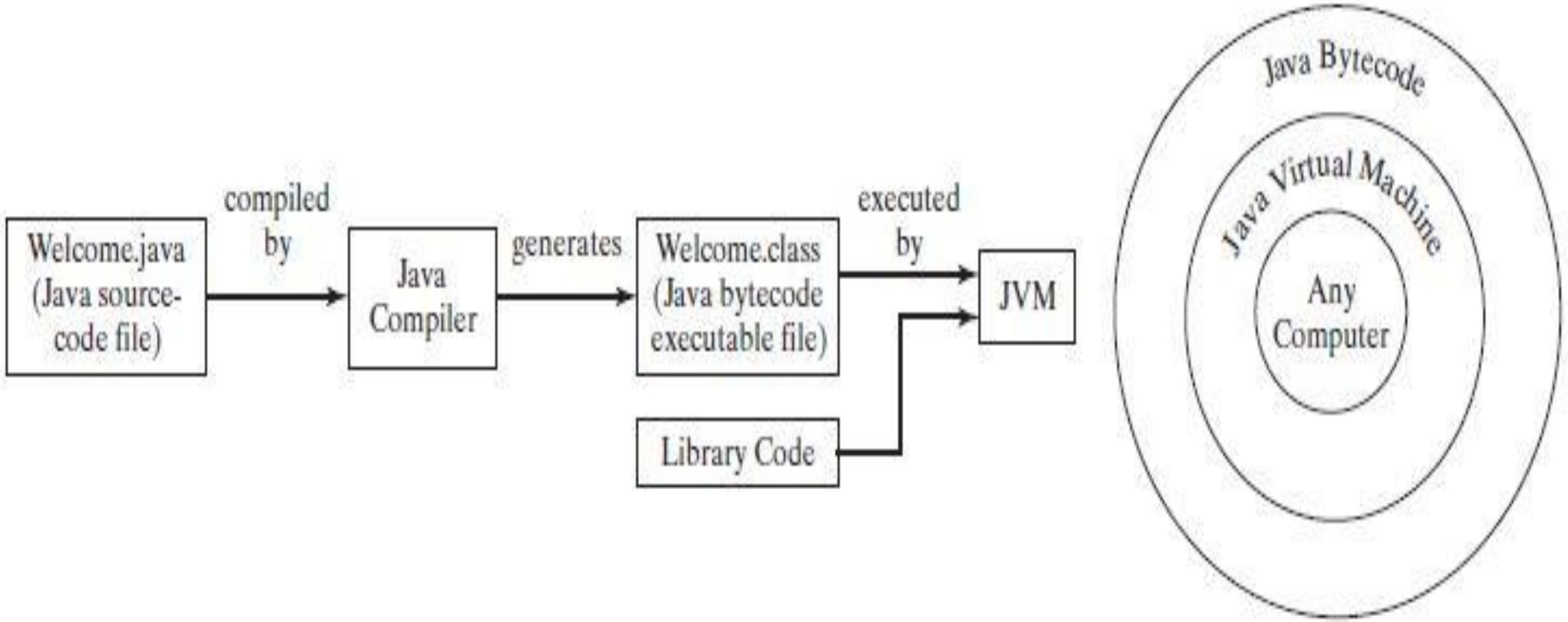
JRE

- JRE is an acronym for *Java Runtime Environment*.
- It is the implementation of JVM and used to provide runtime environment.
- It contains set of libraries and other files that JVM uses at runtime.

Understanding JVM

- JVM (Java Virtual Machine) is an abstract machine.
- It is a specification that provides runtime environment in which java byte code can be executed.
- JVMs are available for many hardware and software platforms.

- The JVM performs following main tasks:
  - Loads code
  - Verifies code
  - Executes code
  - Provides runtime environment


Welcome.java (Java source-code file)
compiled by
Java Compiler
generates
Welcome.class (Java bytecode executable file)
executed by
JVM
Library Code
Java Bytecode
Java Virtual Machine
Any Computer

Internal Architecture of JVM

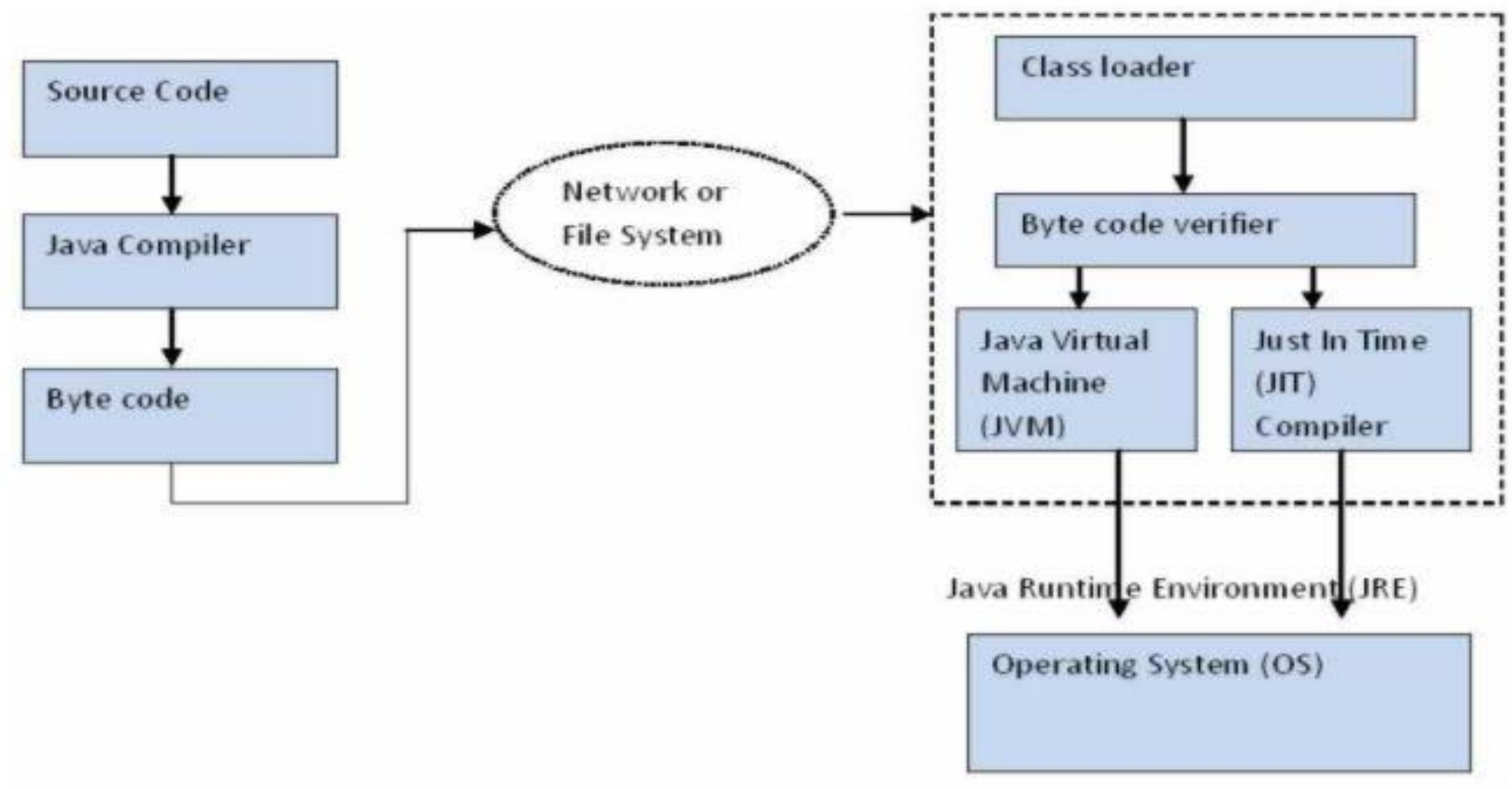
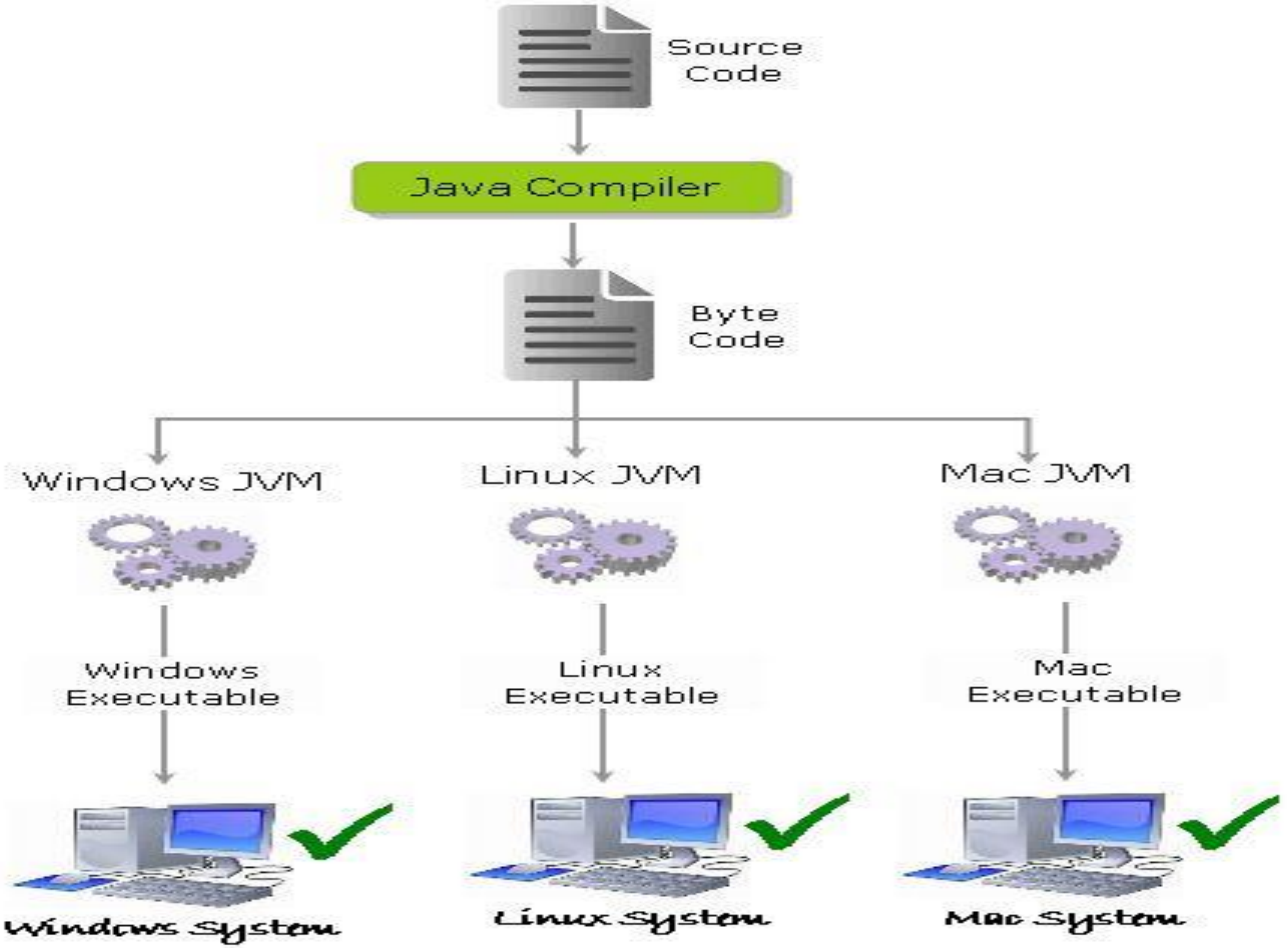


Figure 3: Java Architecture in Detail

How Java is Platform-independent?


Source Code
Java Compiler
Byte Code
Windows JVM
Linux JVM
Mac JVM
Windows Executable
Linux Executable
Mac Executable
Windows System
Linux System
Mac System

How Java is Platform-independent?

- The source code (program) written in java is saved as a file with .java extension.

- The java compiler “javac” compiles the source code and produces the platform independent intermediate code called BYTE CODE. It is a highly optimized set of instructions designed to be executed by the JVM.

How Java is Platform-independent?

- The byte code is not native to any platform because java compiler doesn’t interact with the local platform while generating byte code.

- It means that the Byte code generated on Windows is same as the byte code generated on Linux for the same java code.

- The Byte code generated by the compiler would be saved as a file with .class extension. As it is not generated for any platform, can’t be directly executed on any CPU.

Q1

Which of the following component generates the byte code?

A. javac

B. java

C. javadoc

D. jar

Q2

Java is

A. Compiled
B. Interpreted
C. Compiled as well as Interpreted
D. None of these

Q3

JVM is to generate

A. Bytecode

B. Native code

C. Source code

D. Machine independent code

Q4

What is the extension of bytecode file?

A. .exe

B. .java

C. .class

D. .obj

Java Libraries, Middle-ware, and Database options

- java.lang
- java.util
- java.sql
- java.io
- java.nio
- java.awt
- javax.swing

Sample Java Program

```
import java.lang.*; // Optional

public class Welcome
{
  public static void main(String[] args)
  {
    System.out.println("Welcome to Java!");
  }
}
```

Creating, Compiling and Executing Java Program

- Use any text editor or ~~IDE~~ to create and edit a Java source-code file

- The source file must end with the extension .java and must have exactly the same name as the public class name Welcome.java

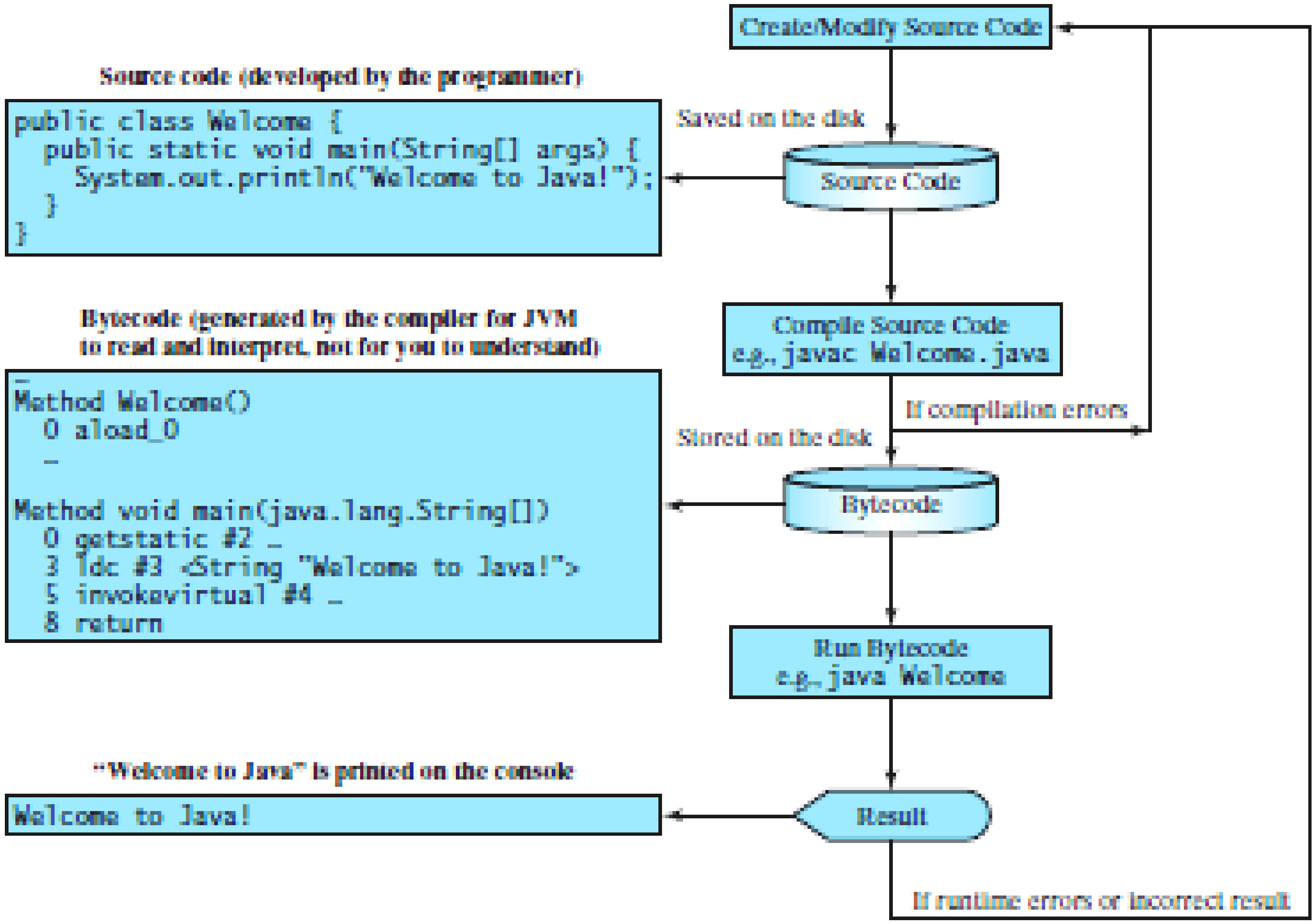
- Compile the .java file into java byte code file (Java bytecode is the instruction set of the Java virtual machine)

  javac Welcome.java

  If there are no syntax errors, the compiler generates a bytecode file with .class extension

- Run the byte code - java Welcome

Creating, Compiling and Executing Java Program



Source code (developed by the programmer)

```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Bytecode (generated by the compiler for JVM to read and interpret, not for you to understand)

```
…
Method Welcome()
  0 aload_0
  …

Method void main(java.lang.String[])
  0 getstatic #2 …
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 …
  8 return
```

"Welcome to Java" is printed on the console

```
Welcome to Java!
```