

MODERN PROGRAMMING TOOLS AND TECHNIQUES-I

Introduction to JAVA

Lecture 1: Genesis and overview of Java

What is JAVA?

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

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.

- Developed by Sun Microsystems (James Gosling)
- A general-purpose Object-Oriented language
- Based on C/C++
- Designed for easy Web/Internet applications
- Widespread acceptance

What is Object Oriented Programming?

Object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of one or more hierarchy of classes united via inheritance relationships.

Principles of OOP

- ▶ All object-oriented programming languages provide mechanisms that help you implement the object-oriented model.
- ▶ They are:
 - ▶ Encapsulation
 - ▶ Inheritance
 - ▶ Polymorphism

Data Encapsulation

Encapsulation is the mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse.

```
class Account {  
    int x=10;  
    float withdraw();  
    void deposit(float amount);  
    float balance;  
}
```

Advantages of Encapsulation

- ▶ Protection
- ▶ Consistency
- ▶ Allows change

Inheritance




- ▶ Inheritance is the process by which one object acquires the properties of another object.
- ▶ A class that is derived from another class is called a subclass (also a derived class, extended class, or child class).
- ▶ The class from which the subclass is derived is called a super class (also a base class or a parent class).
- ▶ A class which is a subtype of a more general class is said to be inherited from it.

Inheritance cont'd

- ▶ The sub-class inherits the base class' data members and member functions.
- ▶ A sub-class has all data members of its base-class plus its own.
- ▶ A sub-class has all member functions of its base class (with changes) plus its own.

Polymorphism

- ▶ Generally, the ability to appear in many forms.
- ▶ In object-oriented programming, *polymorphism* refers to a programming language's ability to process objects differently depending on their data type or class.
- ▶ More specifically, it is the ability to redefine *methods* for *derived classes*.
- ▶ For example, given a base class *shape*, polymorphism enables the programmer to define different *area* methods for any number of derived classes, such as circles, rectangles and triangles. No matter what shape an object is, applying the *area* method to it will return the correct results.

- 
- ▶ Polymorphism allows one interface to be used for a general class of actions.
 - ▶ The specific action is determined by the exact nature of the situation.
 - ▶ One interface
 - ▶ Multiple implementations
 - ▶ Inheritance
 - ▶ Method overloading

Why Java?

11

Java enables users to develop and deploy applications on the Internet for servers, desktop computers, and small hand-held devices.

The future of computing is being profoundly influenced by the Internet, and Java promises to remain a big part of that future.

Where it 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:

1. Desktop Applications such as acrobat reader, media player, antivirus etc.
2. Web Applications such as irctc.co.in, javatpoint.com etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games etc.

Types of Java Applications

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

1) Standalone Application

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

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.

3) Enterprise Application

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

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

Java Platforms / Editions

1) Java SE (Java Standard Edition)

It is a java programming platform. It includes Java programming APIs such as `java.lang`, `java.io`, `java.net`, `java.util`, `java.sql`, `java.math` etc. It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection etc.

2) Java EE (Java Enterprise Edition)

It is an enterprise platform which is mainly used to develop web and enterprise applications. It is built on the top of Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA etc.

3) Java ME (Java Micro Edition)

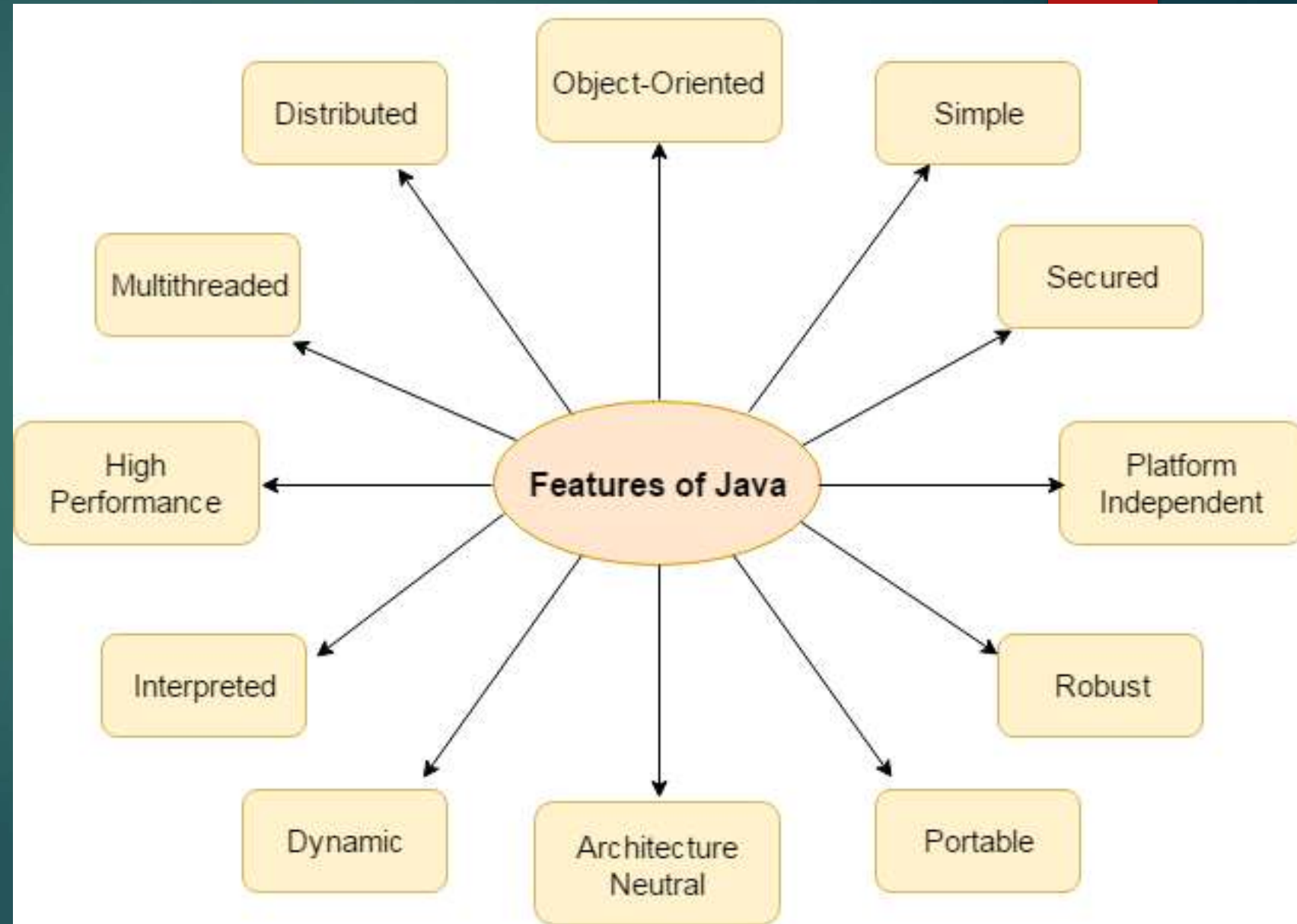
It is a micro platform which is mainly used to develop mobile applications.

4) JavaFx

It is used to develop rich internet applications. It uses light-weight user interface API.

Features of Java

1. Simple
2. Object-Oriented
3. Platform independent
4. Secured
5. Robust
6. Architecture neutral
7. Portable
8. Dynamic
9. Interpreted
10. High Performance
11. Multithreaded
12. Distributed



Simple

According to Sun, Java language is simple because:

- syntax is based on C++ (so easier for programmers to learn it after C++).
- removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc.
- No need to remove unreferenced objects because there is Automatic Garbage Collection in java.

Object-oriented

Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behaviour.

Object-oriented programming(OOPs) is a methodology that simplify software development and maintenance by providing some rules.

Basic concepts of OOPs are:

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

Platform Independent

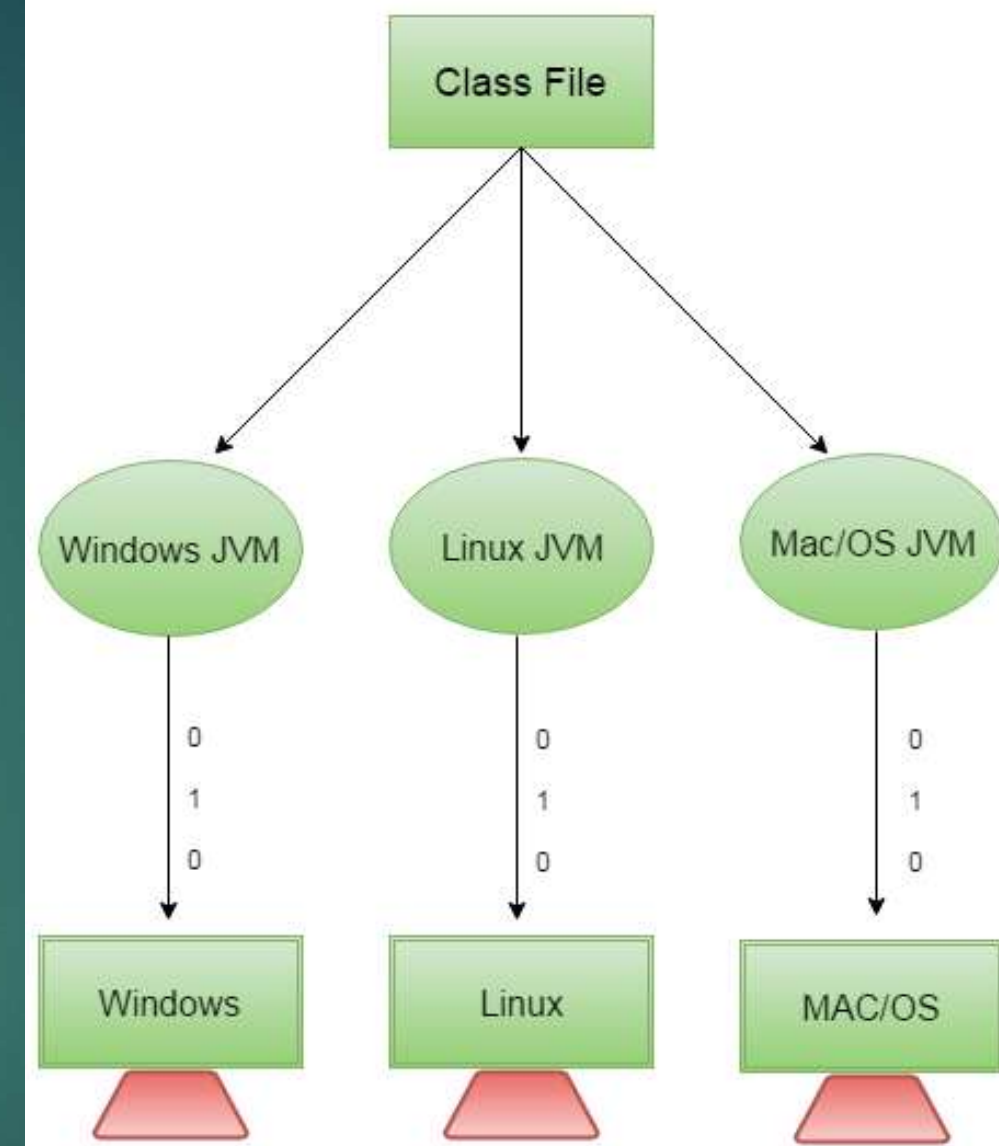
A platform is the hardware or software environment in which a program runs.

There are two types of platforms software-based and hardware-based. Java provides software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

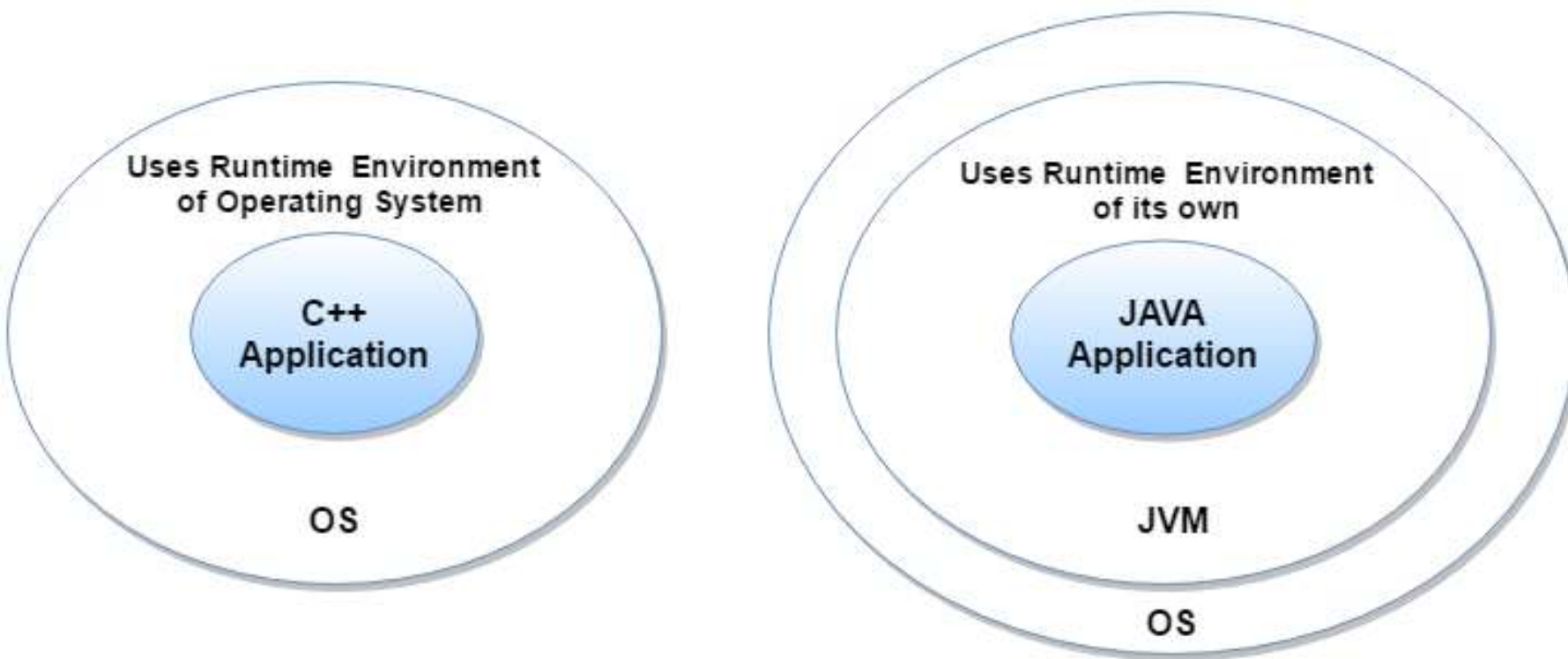
Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere(WORA).



Secured

Java is secured because:

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



- **Classloader:** adds security by separating the package for the classes of the local file system from those that are imported from network sources.
- **Bytecode Verifier:** checks the code fragments for illegal code that can violate access right to objects.
- **Security Manager:** determines what resources a class can access such as reading and writing to the local disk.

Robust

Robust simply means strong. Java uses strong memory management. There are lack of pointers that avoids security problem. There is automatic garbage collection in java. There is exception handling and type checking mechanism in java. All these points makes java robust.

Architecture-neutral

There is no implementation dependent features e.g. 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. But in java, it occupies 4 bytes of memory for both 32 and 64 bit architectures.

Portable

We may carry the java bytecode to any platform.

High-performance

Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++)

Distributed

We can create distributed applications in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet.

Multi-threaded

A thread is like a separate program, executing concurrently. 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.

Difference between Java and C++

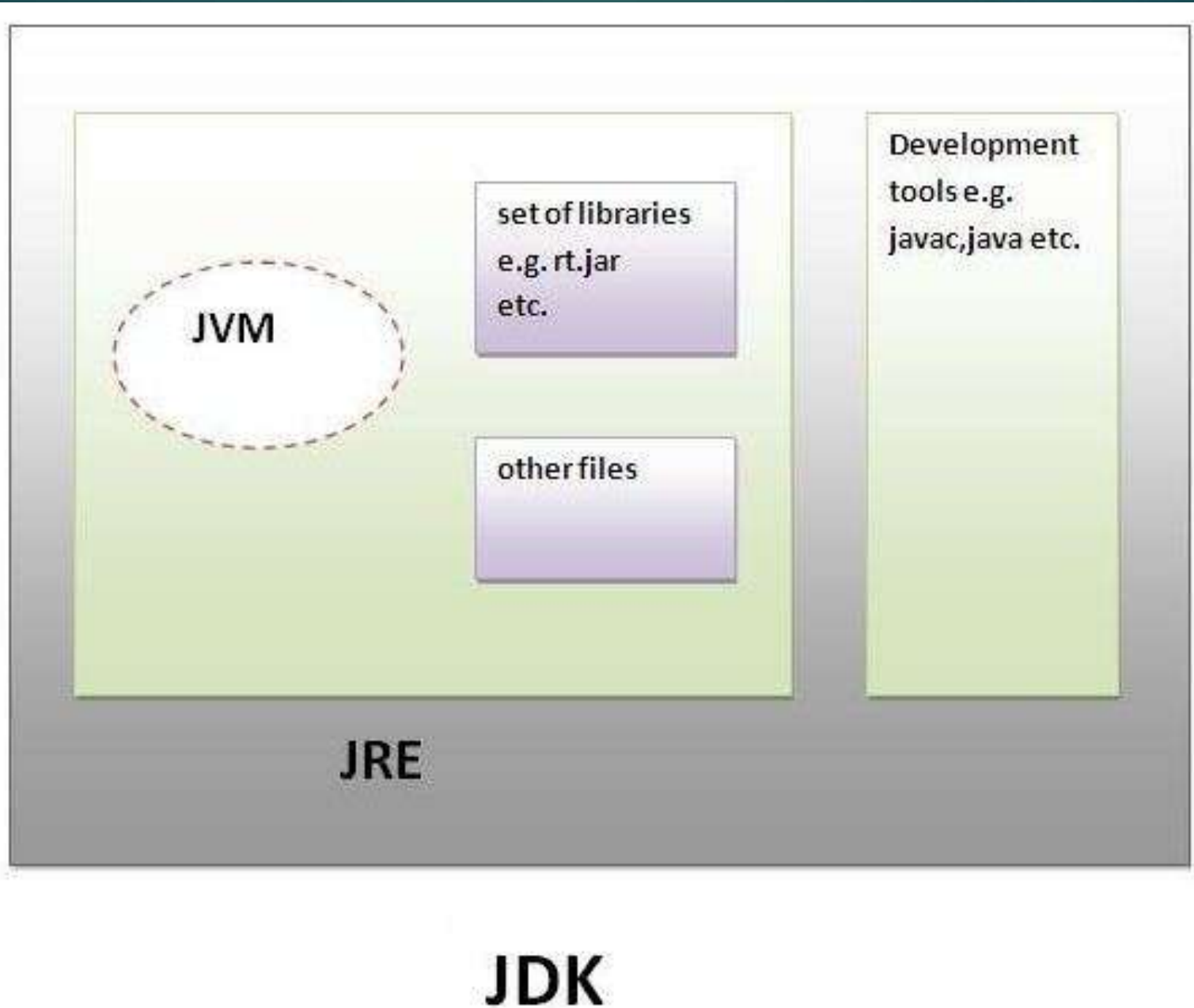
Java

- Single inheritance
- C data type not supported
struct, union, pointer
- Command line arguments →
args
- String → First-class object
- Exception handling →
Try-Catch-Finally
- Garbage collection
- No operator overloading

C++

- Multiple inheritance
- C data type supported
- Command line arguments →
argc, argv
- String → character array
- Exception handling →
Try-Catch
- No garbage collection
- Operator overloading

Understanding JDK, JRE and JVM



JVM

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms. JVM, JRE and JDK are platform dependent because configuration of each OS differs. But, Java is platform independent.

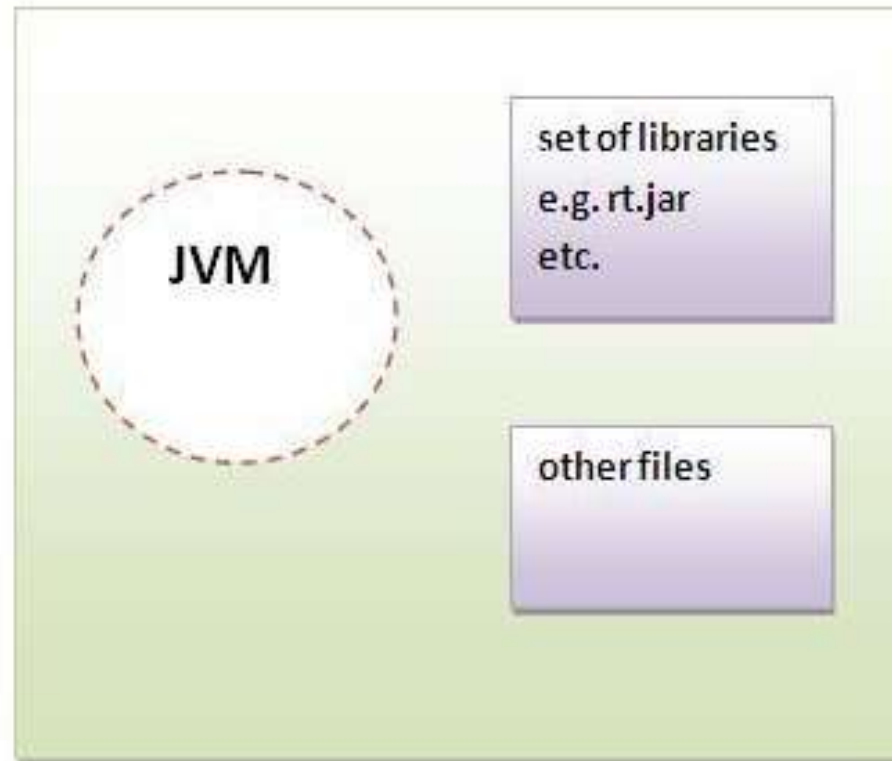
The JVM performs following main tasks:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JRE

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

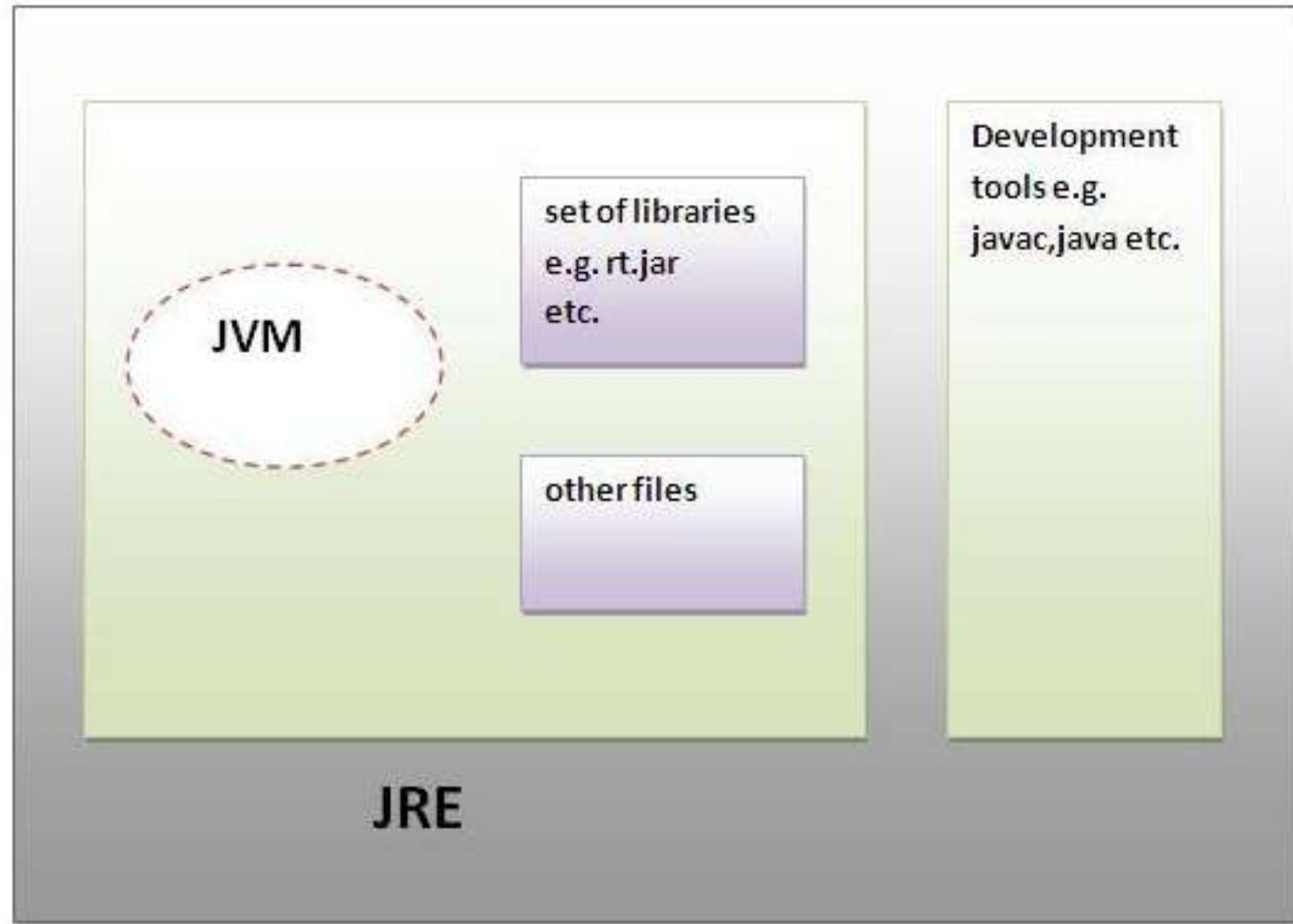
Implementation of JVMs are also actively released by other companies besides Sun Micro Systems.



JRE

JDK

JDK is an acronym for Java Development Kit. It physically exists. It contains JRE + development tools.

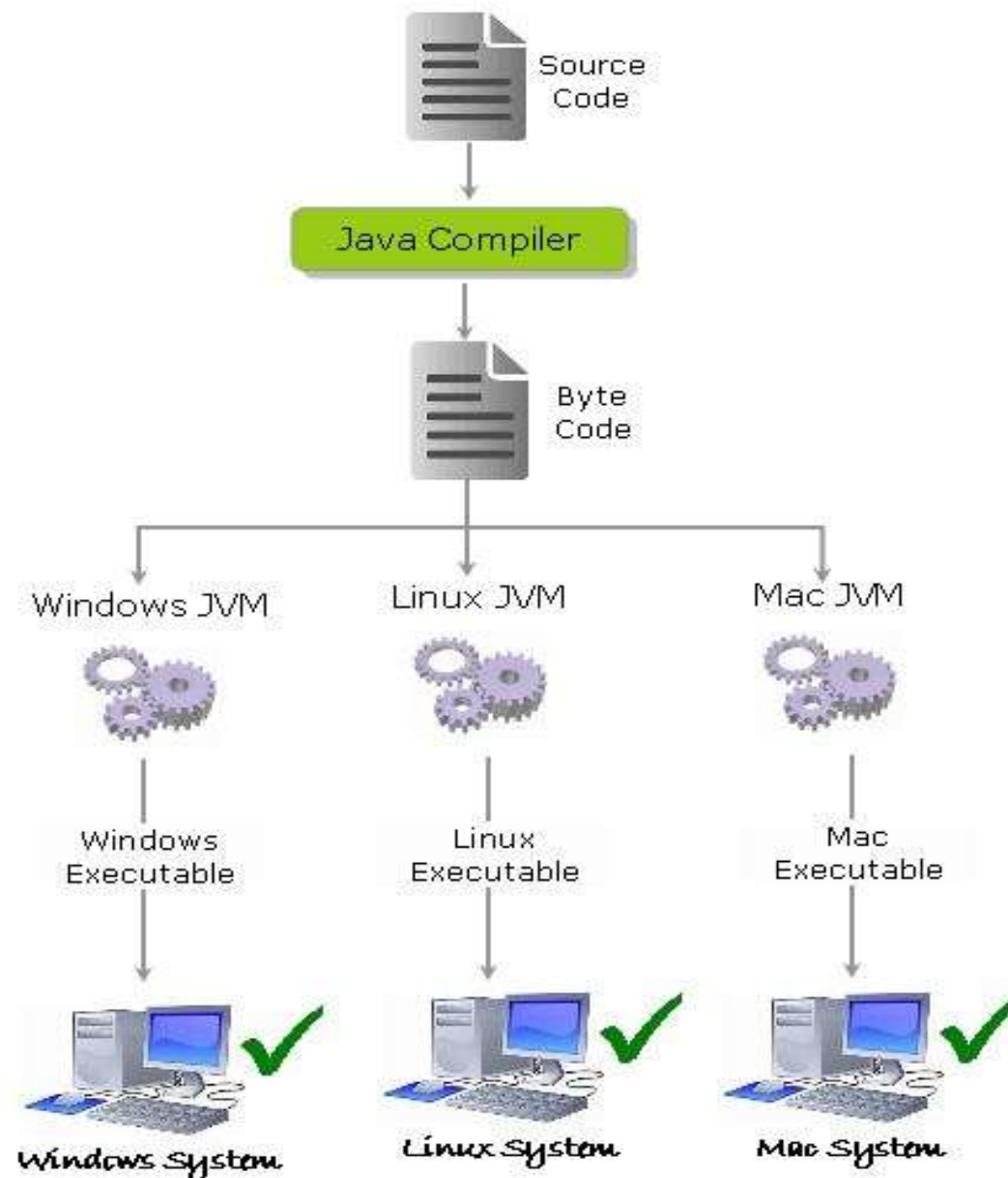


JDK

How Java is Platform-independent?



How Java is Platform-independent?



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.

Creating hello java example

Let's create the hello java program:

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

save this file as Simple.java

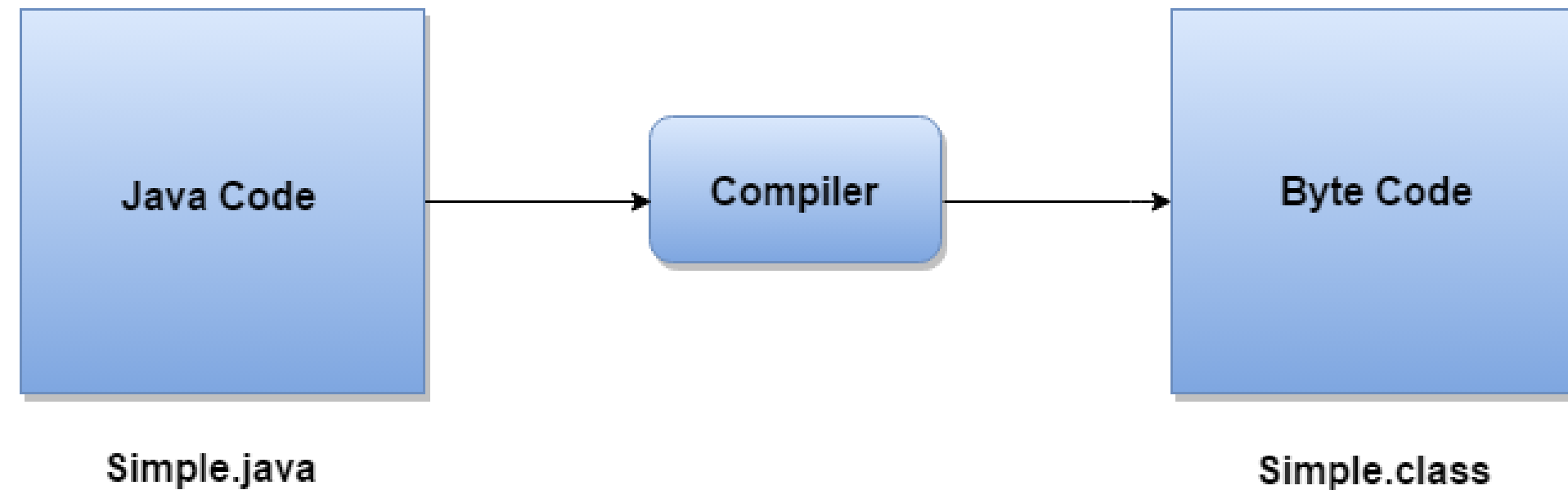
To compile: javac Simple.java

To execute: java Simple

Output: Hello Java

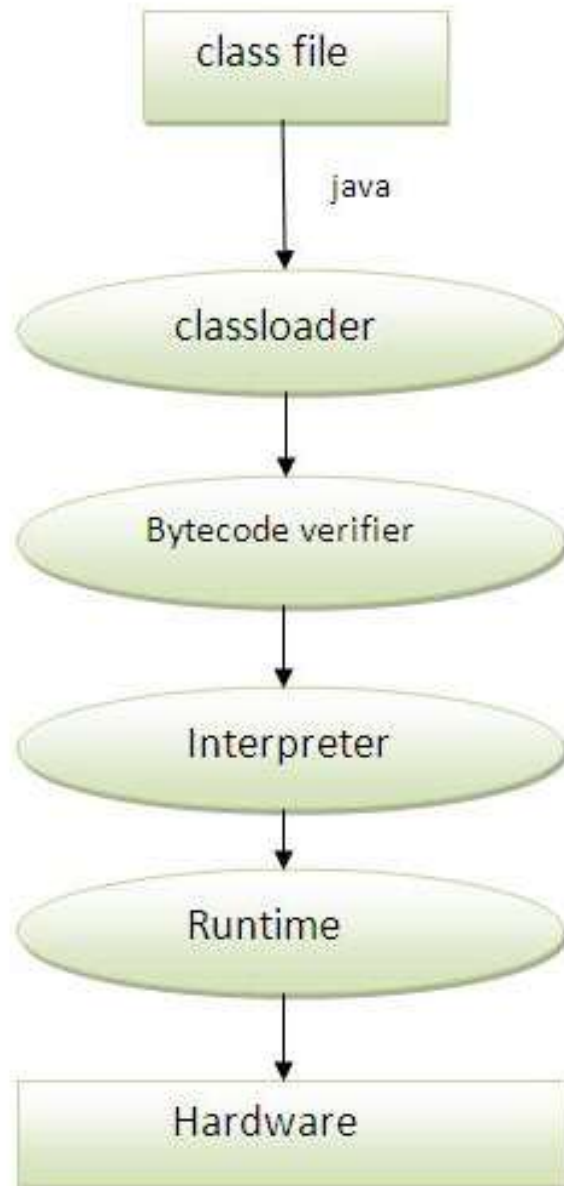
What happens at compile time?

At compile time, java file is compiled by Java Compiler (It does not interact with OS) and converts the java code into bytecode.



What happens at runtime?

At runtime, following steps are performed:

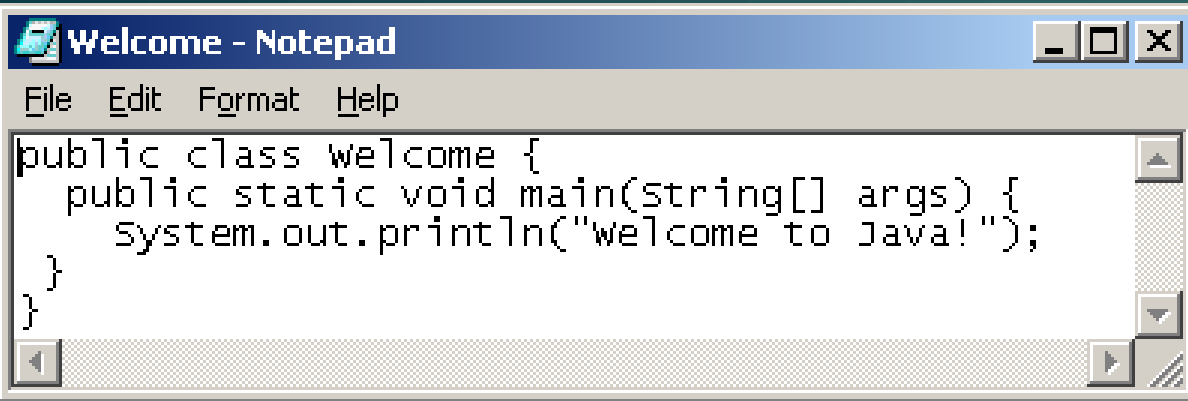


Classloader: is the subsystem of JVM that is used to load class files.

Bytecode Verifier: checks the code fragments for illegal code that can violate access right to objects.

Interpreter: read bytecode stream then execute the instructions.

Creating, Compiling, and Running Programs



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

Source code (developed by the programmer)

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

Byte code (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
```

Create/Modify Source Code

Saved on the disk

Source Code

Compile Source Code
i.e., `javac Welcome.java`

If compilation errors

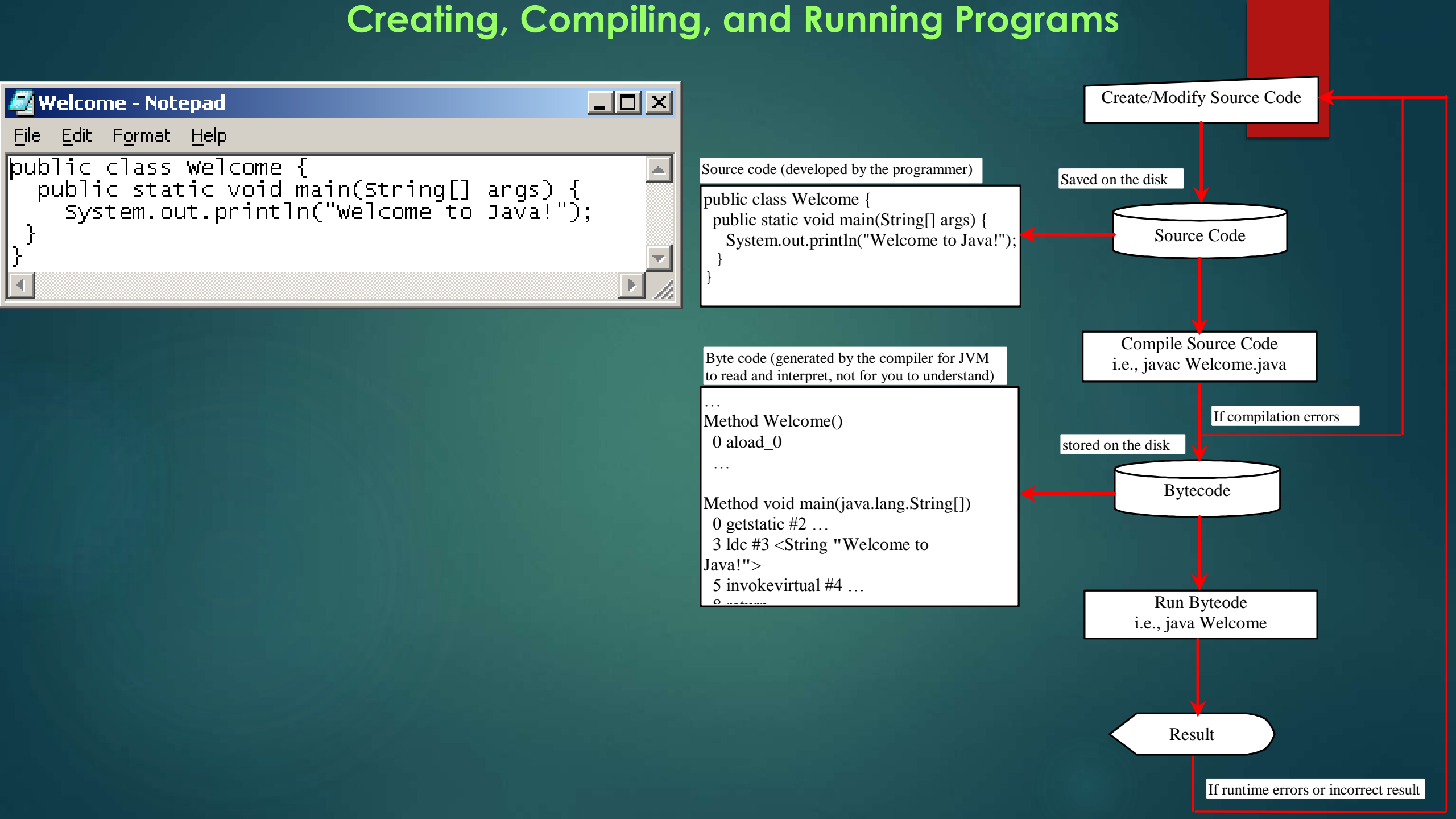
stored on the disk

Bytecode

Run Bytecode
i.e., `java Welcome`

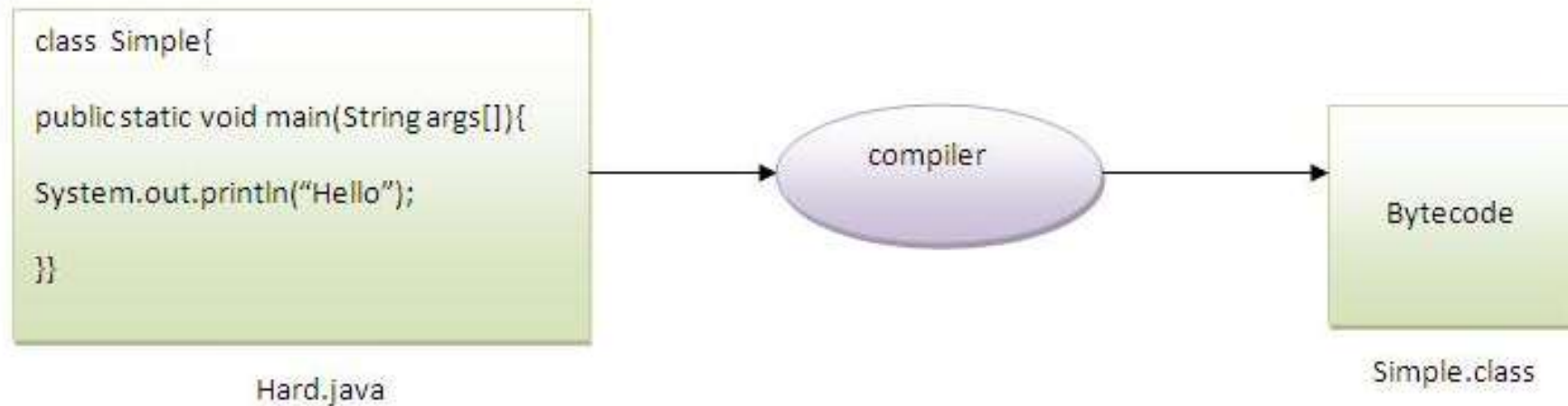
Result

If runtime errors or incorrect result



Q)Can you save a java source file by other name than the class name?

Yes, if the class is not public. It is explained in the figure given below:



To compile: `javac Hard.java`

To execute: `java Simple`

Q)Can you have multiple classes in a java source file?

Yes, like the figure given below illustrates:

