Lab
Principles of Programming Languages

eNotes: CSE/IT/MCA/BCA/MSc

Lab: Principles of Programming Languages

Exp-2

Understand OOP: Java

What is Java?

- Java is a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies.
- It is a computing platform for application development. Java is fast, secure, and reliable.
- It is widely used for developing Java applications in laptops, data centers, game consoles, scientific supercomputers, cell phones, etc.
- When a programmer writes a Java application, the compiled code (known as bytecode) runs on most operating systems (OS), including Windows, Linux and Mac OS.

What is Java used for?

Here are some important Java applications:

- It is used for developing Android Apps
- Helps you to create Enterprise Software
- Wide range of Mobile java Applications
- Scientific Computing Applications
- Use for Big Data Analytics
- Java Programming of Hardware devices
- Used for Server-Side Technologies like Apache, JBoss, GlassFish, etc

History of Java Programming Language

- Here are important landmarks from the history of the Java language:
- The Java language was initially called OAK.
- Originally, it was developed for handling portable devices and set-top boxes. Oak was a massive failure.
- In 1995, Sun changed the name to “Java” and modified the language to take advantage of the burgeoning www (World Wide Web) development business.
- Later, in 2009, Oracle Corporation acquired Sun Microsystems and took ownership of three key Sun software assets: Java, MySQL, and Solaris.

Java Versions

eNotes: CSE/IT/MCA/BCA/MSc

Java Versions	Release Date
JDK Alpha and Beta	1995
JDK 1.0	23rd Jan 1996
JDK 1.1	19th Feb 1997
J2SE 1.2	8th Dec 1998
J2SE 1.3	8th May 2000
J2SE 1.4	6th Feb 2002
J2SE 5.0	30th Sep 2004
Java SE 6	11th Dec 2006
Java SE 7	28th July 2011
Java SE 8	18th Mar 2014
Java SE 9	21st Sep 2017
Java SE 10	20th Mar 2018
JAVA SE 11	25th Sep 2018
JAVA SE 12	19th Mar 2019
JAVA SE 13	17th Sep 2019
JAVA SE 14	17th Mar 2020
JAVA SE 15	15th Sep 2020 (latest Java Version)

Why to Learn java Programming?

- **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

Java Program

- When we consider a Java program, it can be defined as a collection of objects that communicate via invoking each other's methods.
  - **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.
  - **Class** − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.
  - **Methods** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
  - **Instance Variables** − Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

First Java Program

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

public class Main

This line has various aspects of java programming.

- **a.** public: This is access modifier keyword which tells compiler access to class. Various values of access modifiers can be public, protected, private or default (no value).
- **b.** class: This keyword used to declare a class. Name of class (Main) followed by this keyword.

public static void main (String [ ] args)

Its method (Function) named main with string array as an argument.

a. public: Access Modifier

b. static: static is a reserved keyword which means that a method is accessible and usable even though no objects of the class exist.

c. void: This keyword declares nothing would be returned from the method. The method can return any primitive or object.

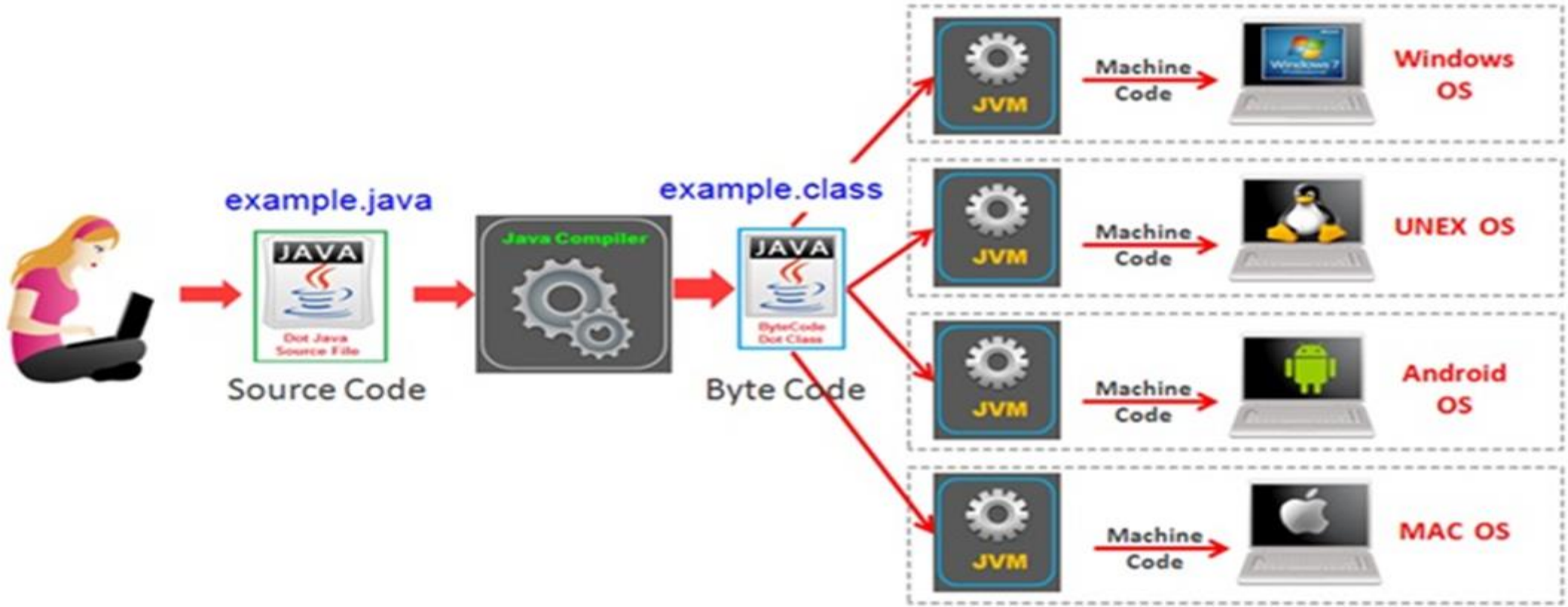
d. Method content inside curly braces. { }

Java Environment

The programming environment of Java consists of three components mainly:

1. **JDK** (Java Development Kit): JDK is intended for software developers and includes development tools such as the Java compiler, Javadoc, Jar, and a debugger.
2. **JRE** (Java Runtime Environment): JRE contains the parts of the Java libraries required to run Java programs and is intended for end-users. JRE can be view as a subset of JDK.
3. **JVM (**Java Virtual Machine) is an abstract machine. It is a specification that provides a runtime environment in which java bytecode can be executed. JVMs are available for many hardware and software platforms.

How Java Program Is Compiled And Works ?



First Java Source Code is Compiled (javac.exe) To Bytecode (.class file)

The JVM Executes the Bytecode (.class file) As Interpreter.

Object-Oriented feature of Java

- Polymorphism
- Inheritance
- Encapsulation
- Abstraction
- Classes
- Objects
- Instance
- Method
- Message Passing

Creating an Object

```
public class Main
{
   void show()
   {
      System.out.println("Welcome to Java");
   }
   public static void main(String[] args)
   {
      //creating an object using new keyword
      Main obj = new Main();
      //invoking method using the object
      obj.show();
   }
}
```