UNIT 1: INTRODUCTION, EVOLUTION AND OVERVIEW OF JAVA LANGUAGE

Lesson Structure

1.0 Objectives

After going through this unit you will be able to:

- Define Java and differentiate between Java and C or C++
- Discuss the features of Java
- Start writing Java programs by setting the local environment for it
- Learn basic concepts important for Java programming.

1.1 Introduction

In this unit we will discuss an overview of Java Programming. Java is an objected oriented programming language developed by Sun Microsystems of USA in 1991. Originally Java was called Oak by James Gosling who was one of the inventors of the language. In 1995 core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]) was released.

The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively. Java is guaranteed to be **Write Once, Run Anywhere.**

1.2 How Java differs from C and C++

(i) Java and C

Java is not lot like C but the major difference between Java and C is that Java is an object-oriented language and has a mechanism to define classes and objects. In an effort to build a simple and safe language, the Java team did not include some of the C features in Java.

1. Java does not include the C unique statement keywords size of and typedef.
2. Java does not contain the data type struct and union.
3. Java does not define the type modifiers keywords auto, extern, register, signed, and unsigned.
4. Java does not support an explicit pointer type.
5. Java does not have a preprocessor and therefore we cannot use # define, # include, and # ifdef statements.
6. Java requires that the functions with no arguments must be declared with empty parenthesis and not with the void keyword as done in C.
7. Java adds new operators such as instanceof and >>>.
8. Java adds labelled break and continue statements.
9. Java adds many features required for object-oriented programming.

(ii) Java and C++

Java is a true object-oriented language while C++ is basically C with object-oriented extension. That is what exactly the increment operator ++ indicates. C++ has

maintained backward compatibility with C. Is is, therefore, possible to write an old style C program and run it successfully under C++. Java appears to be similar to C++ when we consider only the "extensions" part of C++. However, some object - oriented features of C++ make the C++ code extremely difficult to follow and maintain.

Listed below are some major C++ features that were intentionally omitted from Java or significantly modified.

1. Java does not support operator overloading.
2. Java does not have template classes as in C++.
3. Java does not support multiple inheritances of classes. This is accomplished using a new feature called "Interface".
4. Java does not support global variables. Every variable and method is declared within classes and forms part of that class.
5. Java does not use pointers.
6. Java has replaced the destructor function with a finalize() function.
7. There are no header files in Java.

1.3 Features of Java

Java has the following features:

- **Object Oriented** – Java is a pure objected oriented Language. In Java, everything is an Object. Java is based on object model and therefore can be easily extended.
- **Platform Independent** - Java is platform independent that is, it is not compiled into platform specific machine. Java compiler generates byte code instructions that can be implemented on any machine. This byte code is distributed over the web and interpreted by the Java Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple** − Java is easy to learn. Java does not use pointers, preprocessor header files and goto statements. It also eliminates concepts like operator overloading, multiple inheritance which was found in C++.
- **Robust and Secure** – Java is a robust language. It has a strict compile time and run time checking for data types. Java also has the concept of exception handling through which it captures a series of errors and eliminates the risk of crashing the system.

With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

- **Portable** – Java has no implementation dependent aspects which makes it portable. Compiler in Java is written in ANSI C with a clean portability boundary and the size of primitive data types are machine independent.
- **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature support multi-process synchronization and construct interactive applications that can run smoothly.
- **Compiled and Interpreted** – Java combines both compilation and interpretation. First Java compiler translates source code into what is known as bytecode instruction. Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
- **High Performance** – Java architecture is designed to reduce overheads during runtime. Multithreading and Just-In-Time compilers, enables high performance of Java programs.
- **Distributed** − Java is designed for the distributed environment of the internet. It has the ability to share both the data and programs.
- **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

1.4 History of Java

James Gosling initiated Java language project in June 1991 for use in one of his many set-top box projects. The language, initially called ‘Oak’ after an oak tree that stood outside Gosling's office, also went by the name ‘Green’ and ended up later being renamed as Java, from a list of random words.

Sun released the first public implementation as Java 1.0 in 1995. It promised **Write Once, Run Anywhere** (WORA), providing no-cost run-times on popular platforms.

On 13 November, 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).

On 8 May, 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

1.5 How to Setup Local Environment for Java

Follow the instructions to download Java and run the **.exe** to install Java on your machine. Once you installed Java on your machine, you will need to set environment variables to point to correct installation directories −

Setting Up the Path for Windows

Assuming you have installed Java in *c:\Program Files\java\jdk* directory −

- Right-click on 'My Computer' and select 'Properties'.
- Click the 'Environment variables' button under the 'Advanced' tab.
- Now, alter the 'Path' variable so that it also contains the path to the Java executable. Example, if the path is currently set to 'C:\WINDOWS\SYSTEM32', then change your path to read 'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'.

Setting Up the Path for Linux, UNIX, Solaris, FreeBSD

Environment variable PATH should be set to point to where the Java binaries have been installed. Refer to your shell documentation, if you have trouble doing this.

Example, if you use *bash* as your shell, then you would add the following line to the end of your '.bashrc: export PATH = /path/to/java:$PATH'

Popular Java Editors

To write your Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market. But for now, you can consider one of the following −

- **Notepad** − On Windows machine, you can use any simple text editor like Notepad (Recommended for this tutorial), TextPad.
- **Netbeans** − A Java IDE that is open-source and free which can be downloaded from https://www.netbeans.org/index.html.

- **Eclipse** − A Java IDE developed by the eclipse open-source community and can be downloaded from https://www.eclipse.org/.

1.6 Basic elements of Java

When we consider a Java program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do class, object, methods, and instance variables mean.

- **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

Let us look at a simple code that will print the words ***Hello World***.

Example:

```
public class MyFirstJavaProgram {
/* My first Java Program.
*/
public static void main(String []args {
        System.out.println("Hello World"); //prints Hello World
    }
}
```

Next we need to learn how to save the file, compile, and run the program. Please follow the subsequent steps −

- Open notepad and add the code as above.
- Save the file as: MyFirstJavaProgram.java.

- Open a command prompt window and go to the directory where you saved the class. Assume it's C:\.
- Type 'javac MyFirstJavaProgram.java' and press enter to compile your code. If there are no errors in your code, the command prompt will take you to the next line (Assumption : The path variable is set).
- Now, type ' java MyFirstJavaProgram ' to run your program.
- You will be able to see ' Hello World ' printed on the window.

Output

C:\> javac MyFirstJavaProgram.java

C:\> Java MyFirstJavaProgram

Hello World

1.7 Important Points to remember about Java Program

It is very important to keep in mind the following points about Java program:

- **Case Sensitivity** − Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java.
- **Class Names** − For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

  Example: *class MyFirstJavaClass*
- **Method Names** − All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

  Example: *public void myMethodName()*
- **Program File Name** − Name of the program file should exactly match the class name.

  When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match, your program will not compile).

Example: Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as *'MyFirstJavaProgram.java'*

- **public static void main(String args[])** − Java program processing starts from the main() method which is a mandatory part of every Java program.

1.7.1 Java Identifiers

All Java components require names. Names used for classes, variables, and methods are called **identifiers**.

In Java, there are several points to remember about identifiers. They are as follows –

- All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).
- After the first character, identifiers can have any combination of characters.
- A key word cannot be used as an identifier.
- Most importantly, identifiers are case sensitive.
- Examples of legal identifiers: age, $salary, _value, __1_value.
- Examples of illegal identifiers: 123abc, -salary.

1.7.2 Java Modifiers

Like other languages, it is possible to modify classes, methods, etc., by using modifiers. There are two categories of modifiers −

- **Access Modifiers** − default, public , protected, private
- **Non-access Modifiers** − final, abstract, strictfp

We will be looking into more details about modifiers in the next section.

1.7.3 Java Variables

Following are the types of variables in Java −

- Local Variables
- Class Variables (Static Variables)
- Instance Variables (Non-static Variables)

1.7.4 Java Arrays

Arrays are objects that store multiple variables of the same type. However, an array itself is an object on the heap. We will look into how to declare, construct, and initialize in the upcoming chapters.

1.7.5 Java Enums

Enums were introduced in Java 5.0. Enums restrict a variable to have one of only a few predefined values. The values in this enumerated list are called enums. With the use of enums it is possible to reduce the number of bugs in your code.

For example, if we consider an application for a fresh juice shop, it would be possible to restrict the glass size to small, medium, and large. This would make sure that it would not allow anyone to order any size other than small, medium, or large.

Example

Live Demo

```
class FreshJuice {
  enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
  FreshJuiceSize size;
}

public class FreshJuiceTest {

  public static void main(String args[]) {
    FreshJuice juice = new FreshJuice();
    juice.size = FreshJuice.FreshJuiceSize.MEDIUM ;
    System.out.println("Size: " + juice.size);
  }
}
```

The above example will produce the following result −

Output

```
Size: MEDIUM
```

Note − Enums can be declared as their own or inside a class. Methods, variables, constructors can be defined inside enums as well.

1.7.6 Java Keywords

The following list shows the reserved words in Java. These reserved words may not be used as constant or variable or any other identifier names.

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw

throws	transient	try	void
volatile	while		

1.7.7 Comments in Java

Java supports single-line and multi-line comments very similar to C and C++. All characters available inside any comment are ignored by Java compiler.

Example

```
public class MyFirstJavaProgram {

  /* This is my first java program.
   * This will print 'Hello World' as the output
   * This is an example of multi-line comments.
   */

  public static void main(String []args) {
    // This is an example of single line comment
    /* This is also an example of single line comment. */
    System.out.println("Hello World");
  }
}
```

Output

Hello World

1.7.8 Using Blank Lines

A line containing only white space, possibly with a comment, is known as a blank line, and Java totally ignores it.

1.8 Summary

In this unit we have discussed the basic concepts of Java Programming. This unit will help you to develop an insight into Java Programming and you can start writing simple programs in java. We have also discussed some important points that should be kept in mind while writing a Java program. A list of Java key words has also been provided to you in this unit so that you can be accustomed to Java syntax easily.

1.9 Questions

1. Discuss how Java differs from C and C++
2. Describe the features of Java
3. How will you set up the environment for Java?
4. Explain some basic elements of Java.
5. Discuss some keywords of Java.

1.10 Suggested Readings

1. E. Balaguruswammy , Programming in Java, A primer 3e, Tata McGraw Hill, New Delhi.