Coding In Java

Shalini Shekhar

BookMedia Publishing House
Pondicherry, 605001, India

Contents

Java Programming

1. Introduction

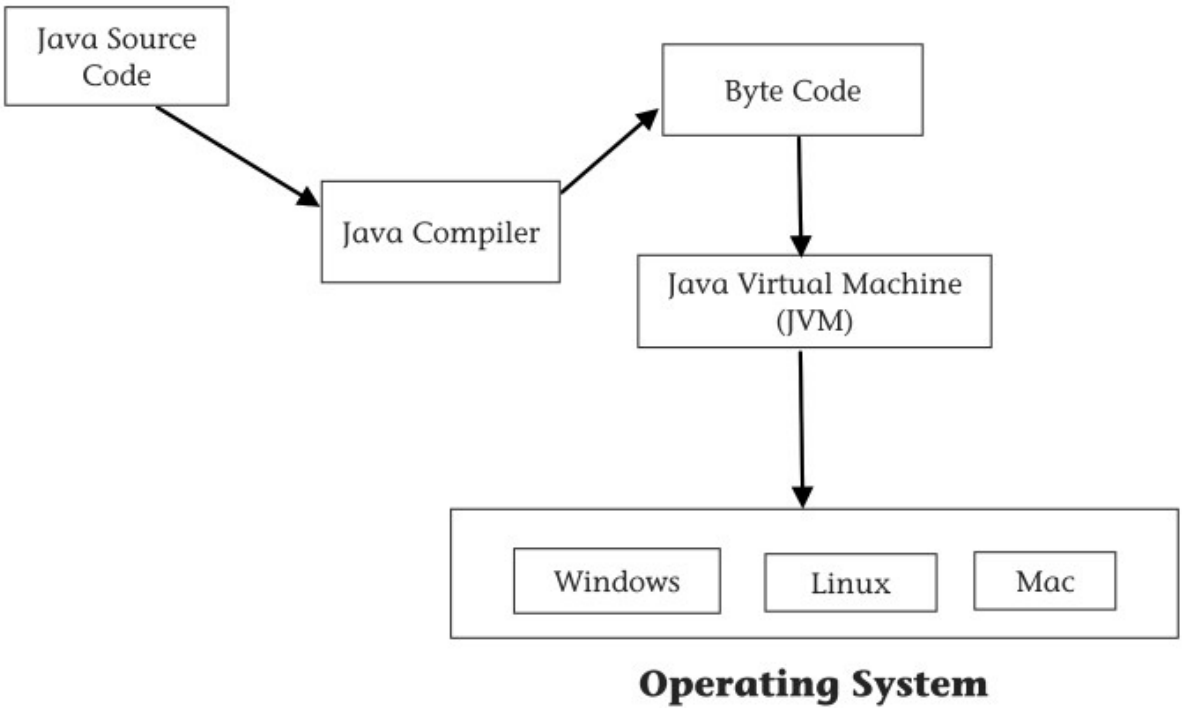
Java is an open source and one of the most popular high level programming language which is easy to learn. Java works on different platform (Windows, Mac, Linux etc.) thus support the concept of "*write once, run anywhere* ". Java is both interpreted and compiled. The java program can be compiled on any platform having a Java compiler. The resulting bytecode (compiled java program) can be run on any hardware platform having Java Virtual Machine(JVM) installed on it.

Java was developed by James Gosling along with his team at Sun Microsystems (acquired by Oracle now)in the year 1995. The language initially called "Oak" after an oak tree and ended up later being renamed as Java.
The following definition of Java by Sun Microsystems lists all the features of Java.

"*Java is a simple, object oriented, distributed, interpreted, robust, secure architecture neural, portable, high performance, multithreaded and dynamic language.* "

Java is commonly used to develop robotics, games, mobile apps, web apps, mobile operating system, embedded systems and much more. There is hardly anything that can't be done using Java.

2. Java Program execution structure



3. Java Components

1. Java Virtual Machine (JVM)

It is a virtual machine which is platform dependent that provides an environment for executing Java bytecodes. JVM interprets the bytecode and convert it into machine code. The main functions of JVM includes

- Loading the code
- Verifying the code
- Executing the code
- Runtime environment.

2. Java Runtime Environment (JRE)

JRE is a software package which provides the runtime environment in which JVM runs. JRE also contains set of libraries classes and the supporting files necessary to run a Java application.

3. Java Development Kit (JDK)

JRE is a part of JDK. Along with Java Runtime Environment JDK also contain various development tools such as appletviewer, java interpreter(java), java compiler (javac), java debugger (jdb) and many more powerful tools for building any Java application.

4. Features of Java

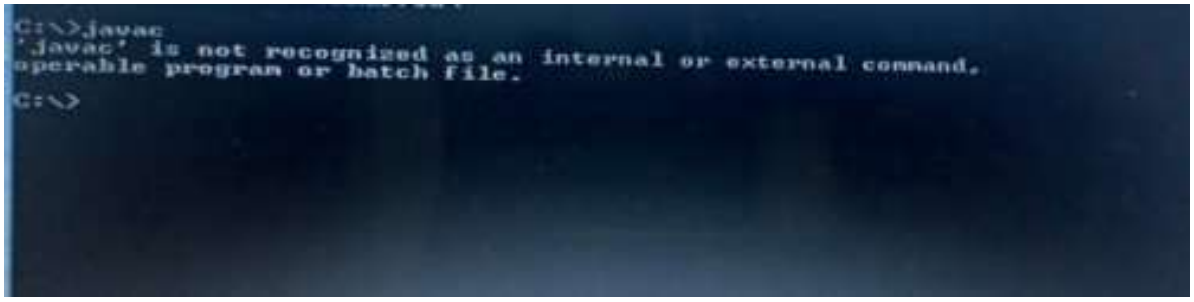
- **Simple** : Java codes are easy to read, learn and write. Its syntax are easy tc understand.
- **Object-Oriented** : Java is purely object oriented and thus provides all OOF features such as abstraction, encapsulation, inheritance and polymorphism. Even the most basic program has a class. Any code in java is written inside a class.
- **Distributed** : Java is designed for the distributed environment of the internet. Java programs can access data across the web as easily as they can do in local system.
- **Compiled and Interpreted** : Java programs are both compiled and interpreted. Java codes is first compiled to bytecode (which can execute on any platform) and then interpreted when java programs are executed.
- **Robust** : Java enables you to handle unexpected errors thus ensuring reliability and bug free programming. It offers automatic garbage collection, handles memory management and different exception handling mechanisms.
- **Secure** : Java ensures security. It provides many safeguard for data against data thieves and uses its own internal memory (instead of pointers) and data management systems to strict any unauthorized data access.
- **High Performance** : Java is faster than other interpreter-based languages.
- **Platform-Intependent and Portable** : Code longevity and portability was a prime concerned in the development of Java language. Java is a platform

independent language that simply means that once java programs are compiled(bytecode), it can run on any platform.“ Write once, runs anywhere” is the motto of Java language.

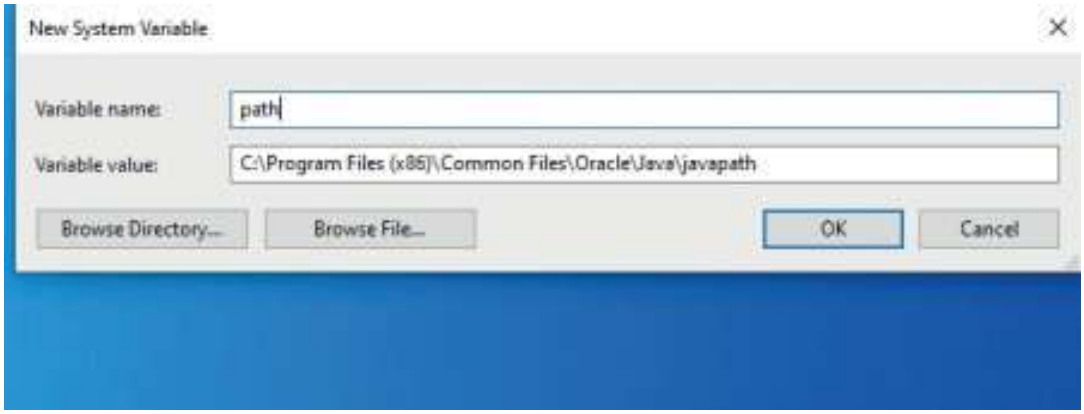
- **Multithreaded** : Java supports multithreaded programming, which allow multiple parts of a program to run simultaneously.
- **Dynamic** : It maintain different versions of an application very easily. It supports dynamic loading of classes.

5. Installing Java

- To check whether Java is installed in your system. Open the command prompt and simply type **javac** and press enter.
- If Java is not installed, following message will be displayed.



- If Java is installed, the following information will be displayed.



6. Close all windows by clicking **OK** .

To write Java programs, we need a text editor. Most commonly used text editors are Notepad (text editor for Microsoft Windows), Netbeans and Eclipse. Netbeans and Eclipse are open source Integrated Development Environment for Java Application development.

7. Objects and Classes

Java is an object oriented programming language, thus every application we develop in java is done by using objects and classes.

An **Object** is an entity that has an identity(name), state (properties) and behaviour (functionality). For example an Apple iPhone has states as ‘width’, ‘height’, ‘color’ and behaviour as ‘to call’, ‘browse internet’, ‘play movie’. An Object is the instance of a class. So, before creating an object in Java, a class should be defined. To access member of any particular class, an object is created of the respective class. An object is created by using the new keyword in java.

A **Class** is defined as a group of objects which have common properties. Identity, state and behaviour of an object are declared in a class. With the help of an object, members of one class can be called from another class. Any number of objects can be created of a class.

To define a class and creating its object.

Syntax:

```
class MobilePhones // defining a MobilePhone class
{
    String name= “Apple iPhone”; // data member
```

```
    String color= “White”; // data member
    int height= 5 ; // data member

    public static void main(String args[])
{
    // creating an object obj of class MobilePhone
    MobilePhones obj=new MobilePhones();

    // accessing its member variable.
    System.out.println(obj.name);
    System.out.println(obj.color);
    System.out.println(obj.height);


}
}
```

Output

Apple iPhone
White
5

8. Java Keywords

Keywords are reserved words that are predefined in Java. Each keywords have special meaning and cannot be used as variable or class name.

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

9. Simple Java Programs

- Open a notepad and write the following code

Example 1: To print a statement

```
public class First
{
    public static void main(String args[])
    {
     System.out.println("This is my first Java Program"); // This is a comment
    }
}
```

Output:

```
This is my first Java Program
```

- Java codes must be written inside a class. class keyword is used to declare a class in java. In the above example we have written our codes in the class named *First* . The first letter of a class always starts with uppercase and it is case sensitive.
- Every Java program has at least one class and one main method.
- The **main()** method is necessary for every java program.
- The code written inside the main() method is executed. The JVM executes everything between the curly braces {} of the main method
- The curly "{" marks the beginning and " }" marks the end of the block of code.
- **System.out.println()** method to print a text on screen where **System** ia a predefined class in java, **out** is a variable declared in the **System** class and **println()** is a method name.
- Each statement must end with a semicolon(;).

9.1 Java main() method

- Java program file (in which the code is written) should be saved with the same name of the class that has the **main()** method.
- In the above case the program is saved with the name First.java.
- public keyword is an access modifier (access modifier discussed later).
- static is a keyword used to declare any method as static. static methods can be called without creating an object of the class.
- void is the return type of the method that returns no value.
- String[] args is command line argument (study later).

9.2 Java Comments

- Single line comment // comment
- Multi-line comment /* comment */

9.3 Java Programs Execution

- Save the file name as First.java. Extension .java is required to save the file because name of the class and filename should match in Java.

10. Variables in Java

Variables are used to store data in Java programs. Variables must be declared before it is used in Java programs specifying what type of data the variable will hold. Variable names are case sensitive. The value of a variable can change during execution of a program.

Example 2: To illustrate variable in java program.

```
public class JavaVariable
 {
   public static void main (String args[])
    {
     int v=5;
     System.out.println("Variable v= " +v);
     v=7;
      System.out.println("After value changed, Variable v= " +v);
    }
   }
```

Output:

```
Variable v= 5
After value changed, Variable v= 7
```

Certain rules and conventions govern the naming of variables.

A variable name :

- Must not be a keyword in Java
- Must not begin with a digit
- Must not contain embedded spaces.

To store a fixed (constant) values which you don't want to change during execution of a program, final keyword is used.

Example 3: To illustrate variable that stores a constant value.

```
public class JavaConstant
 {
  public static void main (String args[])
   {
    final int num=5;
      num=7;
   }
 }
```

Output:

```
JavaConstant.java:7: error: cannot assign a value to final variable num
      num=7;
      ^
1 error
```

11. Data Types in Java

Variables in Java must be declared before they can be used in Java Programs. Data types specify the size and type of values that a variable can take.

There are two types of data types in Java.

i. Primitives Data Types (predefined data types)

ii. Non-Primitive(Derived)

11.1 Primitives Data Types

These are basic data type which are independent of any other data type .

Data type	Size (in bits)	Description	Example	Range	Default values
byte	8	Integer	-100, 126	-128 to 127 (signed)	0
short	16	Integer	-1000, 1400	-2^{15} to $2^{15}-1$	0
int	32	Integer	-4300, 2500	-2^{31} to $2^{31}-1$	0
long	64	Integer	-450000L, 350000L	-2^{63} to $2^{63}-1$	0
float	32	Floating Point	-4.67f, 28.44f	3.4e-038 to 3.4e+038	0.0f
double	64	Floating Point	-234.89, 1000.456	1.7e-308 to 1.7e+308	0.0d
char	16	Unicode Character	'a', 67, '\u0011'	\u0000 to \uFFFF	'\u0000'
boolean	1	true or false			false

i. byte

Example 4 : To illustrate byte data type.

```
class DataTypeByte
 {
   public static void main(String[] args)
    {
       byte b;
      b = 120;
      System.out.println("Value of byte b = " +b);
    }
 }
```

Output:

```
Value of byte b = 120
```

ii. short

Example 5 : To illustrate short data type.

```
class DataTypeShort
 {
    public static void main(String[] args)
     {
        short s;
        s = -150;
        System.out.println( " Value of short s = " +s);
     }
 }
```

Output:

```
Value of short s = -150
```

iii. int

Example 6 : To illustrate int data type.

```
class DataTypeInt
{
   public static void main(String[] args)
    {
       int i;
       i = 2250000;
       System.out.println("Value of int i = " +i);
    }
}
```

Output:

```
Value of int i = 2250000
```

iv. long