UNIT 1

Object Oriented programming and Java

Structure

Learning Objectives

After studying this unit, the student will be able to

- What is a java , Java program , compilation of java program, compilers
- Java linked with internet , platform independent , main features of java
- OOPS, classes and objects, simple example of classes

1.0 Introduction to Java

Java has evolved to be the most predominant and popular general purpose programming language of the current age. Java is a simple, portable, distributed, robust, secure, dynamic, architecture neutral, object oriented programming language. Java was originally developed by James Gosling at Sun Microsystems in 1995.. This technology allows the software designed and developed once for an idealized 'virtual machine' and run on various computing platforms.

Java plays a significant role in the corporate world. Companies of all sizes are using Java as the main programming language to develop various applications/projects worldwide. It has found its use in various sectors including banking, insurance, retail, media, education, manufacturing and so on. E-commerce, Gaming, Mobile, Embedded, Media and many more types of applications are being developed using Java. Organizations are developing mission critical applications using Java technologies. This necessitates the corporations to hire highly skilled java developers. On the other hand it opens the doors for many opportunities for java developers. There is significant demand for java developers with sound technical skills. Now Colleges and Universities all over the world are using it in their introduction courses as well as their junior and senior software engineering courses.

First program

We will learn how to write a simple java program. Creating hello java example is too easy. Here we have created a class named simple that contains only main method and prints a message hello java. It is a simple java program. It is very important to note that the java programs are case sensitive.

For execution of java program

- Install the JDK
- Set path of the bin directory under JDK
- Create and edit the java program by using any text editor such as notepad, msword,etc
- Compile and execute the program

Creating hello java Example

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

Save the file as simple.java

To compile : Javac simple.java

To execute : Java simple

Output : Hello Java

Understanding the first java program

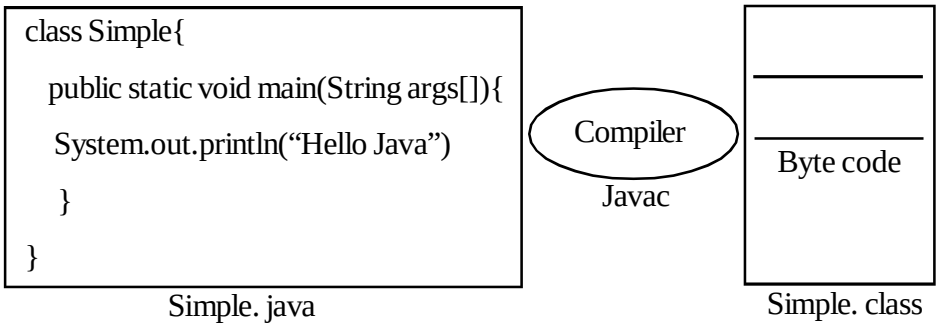
Let us see what are the meanings of class, public, static, void, main, String[ ], System.out.println().

- Class is used to declare a class in java
- Public is an access modifier which represents visibility, it means it is visible to all.
- Static is a keyword , if we declare any method as a static , it is known as static method.

The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it does not require to create an object to invoke the main method. So it saves memory.

- **Void :** Is the return type of the method, it means it doesn't return any value.
- **Main :** Represents the startup of a program
- **String[] args :** Is used for command line argument.
- **System.out.println() :** Is a print statement.

What happens when the program is compiled



At compile time , java file is compiled by java compiler, (it does not interact with OS) and converts the java code into bytecode. Here simple.java will be compiled as simple.class

1.1 Introduction to byte code

Java bytecode is the form of instructions that the Java virtual machine executes. Each bytecode opcode is one byte in length, although some require parameters, resulting in some multi-byte instructions. Not all of the possible 256 opcodes are used. 51 are reserved for future use. Beyond that, Sun Microsystems, the original creators of the Java programming language, the Java virtual machine and other components of the Java Runtime Environment (JRE), have set aside three values to be permanently unimplemented.

A Java programmer does not need to be aware of or understand Java bytecode at all. However, as suggested in the IBM developerWorks journal, "Understanding bytecode and what bytecode is likely to be generated by a Java compiler helps the Java programmer in the same way that knowledge of assembly helps the C or C++ programmer."

1.2 JVM, JRE and JIT Compiler

1.2.1 Java virtual machine (JVM)

A Java virtual machine (JVM) is a virtual machine that can execute Java bytecode. It is the code execution component of the Java software platform.

1.2.2 Java Runtime Environment(JRE)

The Java Runtime Environment (also called **JRE**) is a software framework developed by Oracle Corporation, which runs regardless of the computer architecture. It provides the Java virtual machine and a large library, which allows to run applications written in the Java programming language.

It is constituted by

• A Java virtual machine. The heart of the Java platform is the concept of a "virtual machine" that executes Java bytecode programs. This bytecode is the same no matter what hardware or operating system the program is running under.

• Its associated Java Class Library, which is the Standard library available to Java programs.

1.2.3 JIT compiler

In computing, just-in-time compilation (JIT), also known as dynamic translation, is a method to improve the runtime performance of computer programs based on byte code (virtual machine code). Since byte code is interpreted it executes more slowly than compiled machine code, unless it is actually compiled to machine code, which could be performed before the execution – making the program loading slow – or during the execution. In this latter case – which is the basis for JIT compilation – the program is stored in memory as byte code, but the code segment currently running is preparatively compiled to physical machine code in order to run faster.

1.3 Java and Internet

Java is strongly associated with the internet because of the fact that the first application program written in java was HotJava , a webbrowser to run applets on internet. Internet users can use java to create applet programs and run them locally using a "Java enabled browser" such as HotJava. They can also use Java enabled browser to download an applet located on a computer anywhere in the internet and run it on his local computer.

1.4 Platform independence of java

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 virtual Machine (JVM) on whichever platform it is being run. Platform means an operating system such as windows, unix, xenix, etc.,

It promised Write Once, Run Anywhere (WORA), providing no-cost run-times on popular platforms.

1.5 Introduction to Applets

Applets are small java programs that are primarily used in internet computing. They can be transported over the internet from one computer to another and run using **AppletViewer** or by any Webbrowser that supports java. An applet , like any Application program ,can do many things for us. It can perform arithmetic operations , display graphics , play sounds ,accept user input, create animation ,and play interactive games etc.,

1.6 Main Features of java

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 virtual Machine (JVM) on whichever platform it is being run.

Simple : Java is designed to be easy to learn. If you understand the basic concept of OOP java 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.

Architectural- neutral : Java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors, with the presence Java runtime system.

Portable : We may carry the java bytecode to any platform.

Robust : Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

Multi-threaded : With Java's multi-threaded feature it is possible to write programs that can do many tasks simultaneously. This design feature allows developers to construct smoothly running interactive applications.

Interpreted : 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 : With the use of Just-In-Time compilers Java enables high performance.

Distributed : Java is designed for the distributed environment of the internet.

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.

1.7 Introduction to OOP

Object Oriented Programming or OOP is the technique to create programs based on the real world. Unlike procedural programming, here in the OOP programming model programs are organized around objects and data rather than actions and logic. Objects represent some concepts or things and like any other objects in the real Objects in programming language have certain behavior, properties, type, and identity. In OOP based language the principal aim is to find out the objects to manipulate and their relation between each other. OOP offers greater flexibility and compatibility and is popular in developing larger application. Another important work in OOP is to classify objects into different types according to their properties and behavior. So OOP based software application development includes the analysis of the problem, preparing a solution, coding and finally its maintenance.

Java is a object oriented programming and to understand the functionality of OOP in Java, we first need to understand several fundamentals related to objects. These include class, method, inheritance, encapsulation, abstraction, polymorphism etc.

1.8 Main features of OOPS

There are four main features of OOPS are

1. Abstraction
2. Encapsulation
3. Polymorphism
4. Inheritance

1. Abstraction : The process of abstraction in Java is used to hide certain details and only show the essential features of the object. In other words, it deals with the outside view of an object (interface).

2. Encapsulation : The process of wrapping data into a single unit is called Encapsulation.

3. Inheritance : This is the process in which a properties of a predefined class can be inherited in a new class making the object of that class in the making class.

4. Polymorphism : This is the process in which a program can have more than one function with the same name but different parameters.

The concept of abstraction and encapsulation can be well understood by considering the simple examples. When we drive a motorbike , to stop our bike , simply we press the brakes. When we press the brakes , then the bike will stop. But we don't know the internal mechanism that how it is stopped when we press the brakes because we are **abstracted** from the internal mechanism which a bike is **encapsulated** with a systematic mechanism.

One more example that the mobile phone is **encapsulated** (packaging) with the electronic circuitry with many features such as messages , calls , games, tools, music, internet etc., But to make a phone call , simply we press green button and after our conversation simply we press red button to stop. We know only simple things to start and stop because we are **abstracted** from the internal circuitry of mobile. But there is a large internal process of sending and receiving information through different networks and different places with electronic circuitry.

1.9 Classes and Objects

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. Class is a logical entity and object is physical entity. OOPS simplifies the software development and maintenance by providing some concepts.

- Class
- Object

1.9.1 Class

A class is a group of objects that gave common property. It is a template or blueprint from which objects are created.

A class in java can contain

- Data member
- Method
- Constructor
- Block

syntax of a class

```
class <class_name>{
   data member;
   method;
}
```

Example for a class, suppose there is a program test.java

```
class test
{ public static void main(String args[])
{ System.out.println("Welcome to II CSE Students");
}}
```

The output of the program is

Welcome to II CSE Students

1.9.2 Object

A runtime entity that has state and behavior is known as an object. For example a pen , a car, a man etc are the objects. It can be tangible or intangible(physical or logical). An object has three characteristics.

- **State :** Represents the data of an object
- **Behavior :** Represents the behavior of an object
- **Identity :** Identity is typically implemented via a unique ID. The value of the ID is not visible to the external user , but is used internally by the JVM to identify each object uniquely.

For example Pen is an object, Its name is Parker , color is blue etc., known as its state. It is used to writing is its behavior. Object is an instance of a class. Class is a template or blue print from which objects are created. So object is the instance of a class.

1.10 Classes: public, protected and private

Almost every thing is a class in Java, except for primitives such as ints, chars, doubles, etc. and all classes(reference types according to the lang spec) are derived from Object.

The cornerstone of most Object Oriented programming languages are classes.

Java has four types of access levels and one unamed default one

- **Public :** Accessible everywhere.
- **Private :** Accessible only within the class
- **Protected :** Accessible to those within the same file(package) and/ or derived classes.
- **Private protected :** Accessible to those within the class and derived classes.

The default access level is very similar to protected.

1.11 Simple example of objects and classes

In this example , we have created a student class that have two data members empid and empname. We are creating the object of the student class by new keyword and printing the objects value.

```
class Student{
int empid; // data member or instance variable
String empname; // data member or instance variable
public static void main(String Args[]){
Student s1 = new Student(); // creating an object of student
System.out.println(s1.empid +"  " +s1.empname);
}}
```

output

0 null

Instance variable

A variable that is created inside the class is also known as instance variable. These variables are only declared and therefore no storage space has been created in the memory. Instance variables are also known as member variables.

Method

In java , a method is like a function i.e, used to expose behavior of an object.

Advantage of method

- Code reusability
- Code optimization

New keyword

The new keyword is used to allocate memory at runtime.

Example of object and class that maintains the records of students

In this example , we are creating the three objects of Student class and initializing the values to these objects by invoking the InsertRecord method on it. Here , we are displaying the state(data) of the objects by invoking the DisplayInformation method.

```
class Student{
int rollno;
String name;
void InsertRecord(int r , String n) {  //method
rollno  = r ;
name  = n;
}

void DisplayInformation(){
System.out.println(rollno+" "+name); }

public static void main(String args[]){
Student s1 = new Student();
Student s2 = new Student();
Student s3 = new Student();
s1.InsertRecord(101,"Rama");
s2.InsertRecord(102,"Surya");
s3.InsertRecord(101,"Chandra");
```

```
s1.DisplayInformation();
s2.DisplayInformation();
s3.DisplayInformation();
  }
 }
```

The output of the above program is

101 Rama

102 Surya

101 Chandra

Another example of object and class

```
class Rectangle{
int length;
int width;

void insert(int i ,int w){
length = i;
width = w; }

void CalculateArea(){
System.out.println("Area = "+length*width);}
public static void main(String args[]){
Rectangle r1 = new Rectangle();
Rectangle r2 = new Rectangle();
r1.insert(12,3);
r2.insert(15,4);
r1.CalculateArea();
r2.CalculateArea();} }
```

The output of the above program is

Area = 36

Area = 60

The first step in the installation of Java is to download the Java Development Kit(JDK) from the **java.sun.com** website. It is free software to download. We must choose the right version of JDK to the configuration of your system (operating system , 32 or 64 bit etc.,) at the time of download . once the setup file is downloaded , we may proceed to install Java.

1.12 Installing and configuring Java

To install Java, we need to perform the following steps

1. Double click the .exe file to initiate the installation procedure. The screen appears as shown in Fg 1.1.

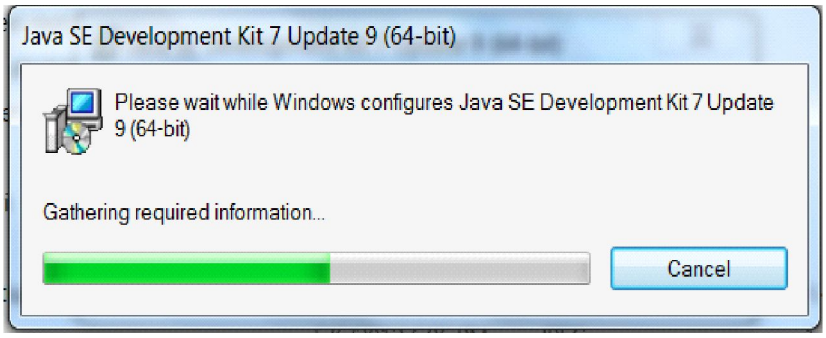


Fig 1.1

2. After then , the welcome screen appears as shown Fig 1.2

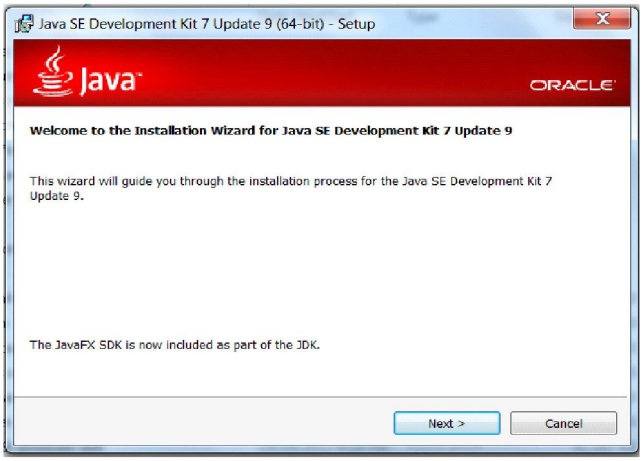


Fig 1.2

3. In the welcome screen , if we click on Next, starts the installation of java. Once the installation is completed , the complete screen appears as shown in Fig 1.3

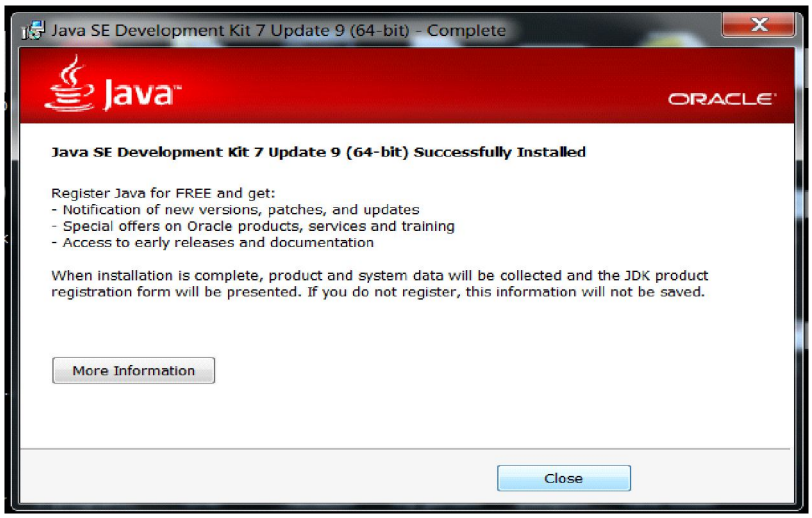


Fig 1.3

4. The screen describes the successful installation of Java on the computer system.

Click the close button to end the installation process.

Configuring Java

Once the Java is installed , we need to configure it by adding the java path to the environment variable, PATH. This eliminates the restriction to use java only from its current working directory. To set the PATH variable to the java directory , we need to perform the following steps.

1. Right click on the "My Computer" or "Computer" icon as shown in Fig 1.4

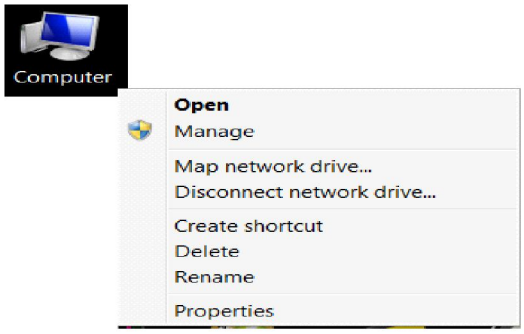


Fig 1.4

And select the properties option from the drop down menu. The system properties dialogue box appears as shown in Fig 1.5

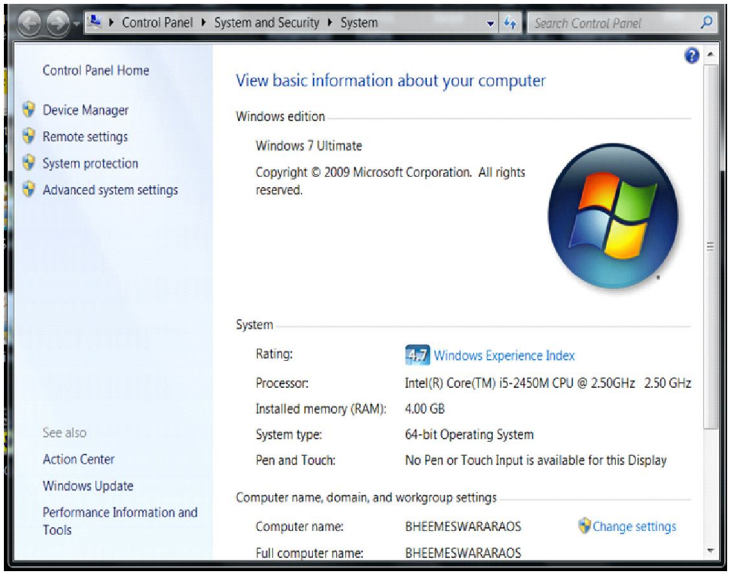


Fig 1.5

Select the "advanced system settings" option to display the advanced page as shown in Fig 1.6

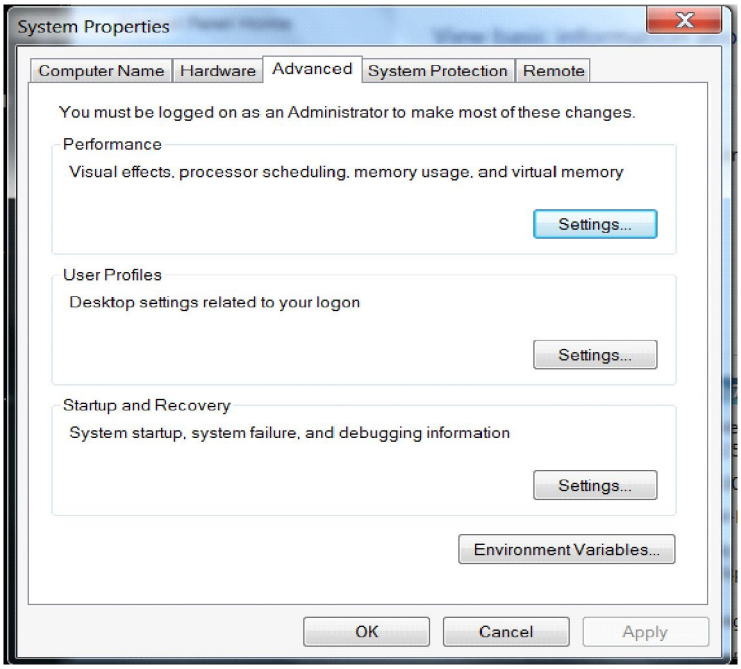


Fig 1.6

Click the “Environment Variables” button to display the Environment variables dialogue box as shown in Fig 1.7

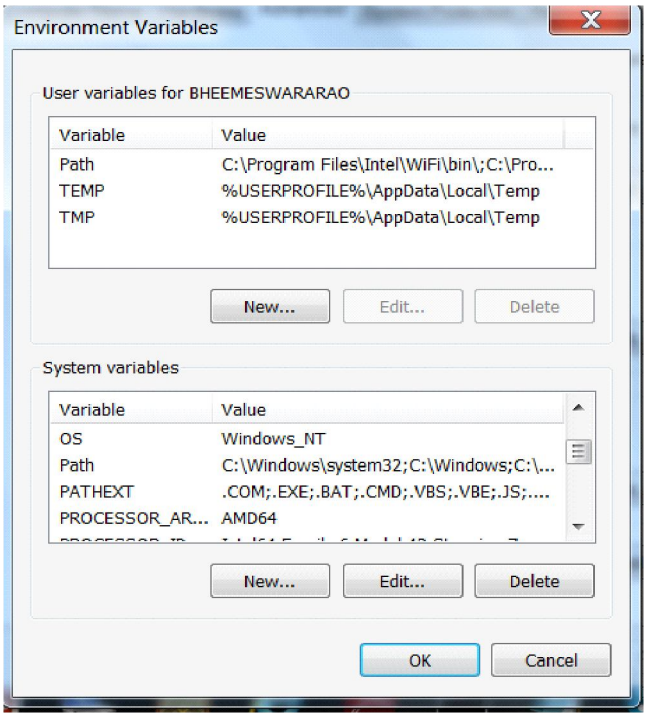


Fig 1.7

The environment variable dialogue box is divided into two sessions

- User variables
- System variables

Under system variable section , select the path option below the variable column and click the Edit button. The Edit system variable dialogue box appears as shown in Fig 1.8

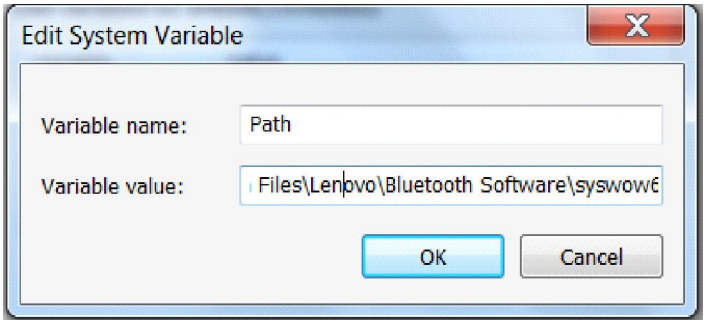


Fig 1.8

By default , the path variable is already set to multiple locations. To set the Java directory path to the Path variable , append the directory path in the variable value text box , separated by a semi-colon ,as shown in Fig 1.9

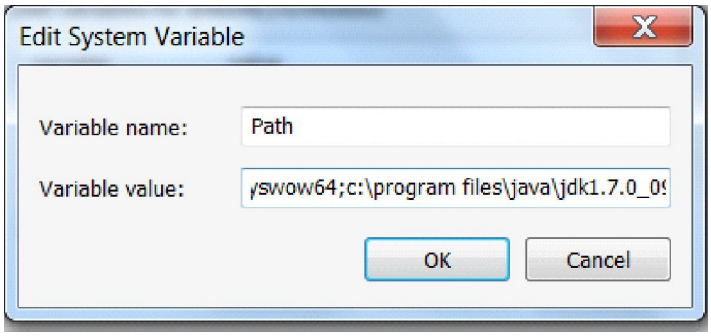


Fig 1.9

click OK to close the Edit system Variable dialogue box.

Click OK to close Environment Variable dialogue box

Click OK to close System Properties dialogue box and complete the process of configuring Java.

Summary

Java programs are case sensitive. Lowercase characters are not equivalent to uppercase characters and vice versa. Suppose a class name CalculatingArea is not equivalent to calculatingarea.

Short Answer Type Questions

1. Who developed java language.
2. Write a simple java program to display “hello world”.
3. What are the commands for compilation and execution of java programs ?
4. What is java bytecode ?
5. What is JVM ?
6. What is JRE ?
7. What is JIT compiler ?
8. How java is platform independent.
9. What are applets ?
10. What is a class ?

11. What is an object ?

12. What is an instance variable ?

13. What are advantages of a method ?

14. What is the expansion of WORA ?

Long Answer Type Questions

1. Explain briefly class,public,static,void,main,string[] and system.out.println().

2. Write about the main features of java.

3. Write about the main features of OOPS.

4. A Record contains employee number(empno) , employee name(ename) and salary(sal) So, write a program to display the details of three employees.

UNIT 2

The Java Programming Language

Structure

Learning Objectives

After studying this unit, you will be able to understand

- Various data types
- Operators in java
- Creating methods and calling them
- Conversion of data types and casting
- Constructors and destructors

2.0 Introduction

Java Tokens: A Java program is basically a collection of classes. A class is defined by a set of declaration statements and methods containing executable statements. Most statements contain expressions ,which describe the actions carried out on data. Smallest individual units in a program are known as tokens.

2.1 Tokens

In simple terms , a Java program is a collection of

(a) Tokens ,

(b) Comments and

(c) White spaces.

(a) Tokens

Java language includes five types of tokens.

1. Reserve Word or Keywords
2. Identifier
3. Literals
4. Operators
5. Separators

1.Reserved words(Keywords)

There are certain words with a specific meaning in java which tell (help) the compiler what the program is supposed to do. These Keywords cannot be used as variable names, class names, or method names. Keywords in java are case sensitive, all characters being lower case.

Keywords are reserved words that are predefined in the language; see the table below (Taken from Sun Java Site). All the keywords are in lowercase.

Abstract	Default	If	Private	This
Boolean	Do	Implements	Protected	Throw
Break	Double	Import	Public	Throws
Byte	Else	Instanceof	Return	Ransient
Case	Extends	Int	Short	Try