MODULE 1

Introduction to Java Programming

Introduction to Object Oriented Programming in Java

One of the most important concepts of a programming language is its reliability. Object Oriented Programming concept helps keeping a programming language to utmost reliable and dependable. Let's look down upon history of Java and how does it come to existence as a programming language after C and C++.

(i) Evolution of Java

Researchers were thinking of a programming language which could be securable, reusable and support all platforms and also acts as a class-based, object-oriented with specifically designed dependencies to fix the drawbacks of existing languages like C and C++. Where C is not secured and C++ does not have the advanced characteristic.

It intended to develop an application program which can be "write once and run anywhere" (WORA), which means the compiled Java code can run on all platforms that support Java.

History of Java

In June 1991, James Gosling, Mike Sheridan and Patrick Naughton initiated the Java language project. The language was initially called Oak after an oak tree that stood outside Gosling's office. Later the project went by the name Green and was finally renamed Java, from Java Coffee.

Java (programming language) Java is a programming language. James Gosling first developed it at Sun Microsystems, which is now a part of Oracle Corporation. It was released in 1995 as a part of Sun Microsystems' Java platform.

The most popular programming languages are C, C++, C#, and Java. Where Java is an advanced programming language and facilitates advanced technical features when compared with each other.
A revolutionary task to develop a program for digital devices such as set top boxes and television was an interesting fact which made Java programming to come into existence. It was an advance concept at that time for Green team members (developers of Java programmers) but it satisfied the demands of internet programming. Later, Netscape incorporated Java programming.

Java is a propertied language which has its best elements to inherit and combine with any innovative concept which can build its own unique mission. Java is a technically blended language which has a rich elements that can address any changing expectations and technical innovations.

At present, Java is one of the most popular programming in use, particularly in client-server architecture, internet programming, mobile devices, games, e-business solutions etc.

Java is case sensitive or not?

Yes, Java is case-sensitive language. It is this way because of its heritage from C language. To keep the language more familiar to what people were used to "in the day", they left it as case-sensitive. There is an added advantage since Java identifiers can be almost any Unicode character in the programs.

Unicode is a universal international standard character encoding that is capable of representing most of the world's written languages such as American Standard Code for Information Interchange (ASCII) for the United States, ISO 8859-1 for Western European Language, GB18030 and BIG-5 for Chinese and KOI-8 for Russian and so on.

Features of Java:

The java features given below are simple and easy to understand. They are also known as java buzzwords.

1. **Simple:** Java is Simple to write and more readable and eye-catching. It has a concise, cohesive set of features that makes it easy to learn and use.

   Syntax of Java is based on C++ (so Java is easier for programmers to learn it after C++).

   Many confusing and/or rarely-used features are removed. For example, explicit pointers, operator overloading etc.

2. **Secure:** Java program cannot harm another system thus making it safe. It provides a reliable means of creating Internet applications and a secure way to access web applications.

   No explicit pointer.

   Java Programs run inside virtual machine sandbox.

3. **Portable:** Java programs can execute in any environment for which there is a Java run-time system (JVM). Java programs OR codes can be run on any platform means OS (Linux, Window, Mac). Java programs can be shifted over World Wide Web (e.g., applets)

   We may carry the java byte code to any platform.

4. **Object-oriented:** Java programming is an object-oriented programming language. Like C++ Java provides most of the object oriented features. Java is pure OOP Language.

5. **Robust:** Java encourages error-free programming by being strictly typed and performing run-time checks.

   There is lack of pointers that avoids security problem, automatic garbage collection, exception handling and type checking mechanism in java. All these points make java robust.

6. **Multithreaded:** Java is a multi-threaded programming language which means we can develop multi-threaded program using Java. A multi-threaded program contains two or more parts that can run concurrently and each part can handle different task at the same time making optimal use of the available resources. It is said to be advanced to C++ because C++ does not contain any built-in support for multithreaded applications. Instead, it relies entirely upon the operating system to provide this feature.

7. **Architecture-neutral:** Now talking about C, C++ they are a platform dependent language's as the file which compiler of C, C++ forms is an .exe (executable) file which is operating system dependent. The C++ program is controlled by the operating system whereas, the execution of a Java program is controlled by JVM (Java Virtual Machine).

8. **Interpreted:** The interpreted feature in C++ is very slow because it is interpreted line by line, and makes the system a slower way of running a program than one that has been compiled.

9. **High performance:** Since C++ is platform dependent, the performance of C++ is not that much noted. Byte codes are highly optimized. JVM can execute them much faster.

10. **Distributed:** Java was designed with the distributed environment. Java can be transmitted, run over the internet. RMI and EJB are used for creating distributed applications.

11. **Dynamic:** Java programs carry with them large amounts of run-time type information that is used to confirm and resolve accesses to objects at run time.

Object Oriented Programming Concepts

Java is an Object Oriented language. It provides the feature to implement an Object Oriented model, where it is organised around objects and logic rather than data it is used at the beginning of software lifecycle and also used to develop other applications.

Object Oriented Programming is a technique in which programs are written on the basis of objects ***for example,*** (Java) where it adopts a concept that views everything as an object and implements a logic about it. Let's see the concepts of Object Oriented Programming in detail with a real time example.

Object

An object is nothing but a matter of thing what we find in everyday life. Technically, Java is a software bundle of related state and behaviour.

An object has a state or properties (information such as size, height, colour, position, etc.) ***for example,***

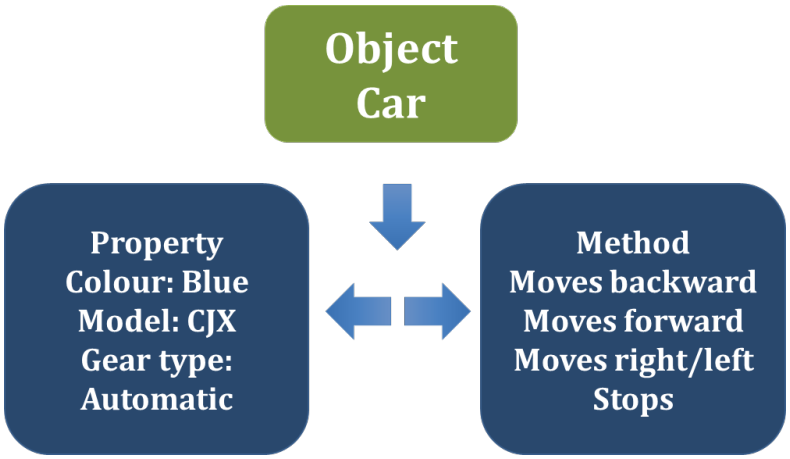


Figure 1.1.1: Object with its Attributes

Class

In Java programming, the class is defined as a blueprint from which objects are created. It models and describes the state and behaviour of an object.

For example, consider a car with several properties like the clutch, steering, break, etc.

Inheritance

Inheritance is a property by which state and behaviour of one class (super class) can be inherited to its sub-class. It is a powerful and a natural mechanism for organising and structuring a software.

For example, Human heredity

Abstraction

Hiding the implementation details from the user is called as the abstraction. Here, only the functionality will be provided. In Java, abstraction can be achieved by the abstract class and Interface.

A class with an abstract keyword is called as abstraction.

Interface – blueprint of a class. Full abstraction can be achieved using this mechanism.

For example, Invention of flight is based on the flying mechanism learnt from birds. So flights are derived from base class birds.

Encapsulation

Encapsulation is one of the fundamental concept of OOP. Mechanism of binding the data and function together in a single unit is called as encapsulation.

For example, Medicine hidden by a capsule. Where medicine is an important component which is hidden by a capsule.

Polymorphism

Polymorphism is an ability of an object to take many forms. It reacts differently in different situation. In Java, polymorphism is used to reduce code complexities. Methods or function of a class implies the behaviour of polymorphism. In Java, all Java objects are polymorphic.

For example, Frogs live indifferently on land or in water.

(ii) Components of Java

Java Compiler

The Compiler is used to systemise the program and check against language syntax rules. It converts the source code to byte code and then compiles them. The Java compiler generates an architecture-neutral object file format (the compiled code – source code and byte code) to be executable on many processors.

Source code: Collection of computer instructions written in human readable format.

- Write in java class files

Byte code: Computer object code, also called as instruction set of Java Virtual Machine.

- Writes the output in class file

Java Virtual Machine (JVM)

Java Virtual Machine – enables the computer to run Java program. It reads and interprets .class files and executes the program's instruction into the native hardware platform. JVM is platform independent (obtainable for many hardware and software platforms). In the Java virtual machine specification, the behaviour of a virtual machine instance is described in terms of subsystems, memory areas, data types and instructions. These components describe an abstract inner architecture for the abstract Java virtual machine. The purpose of these components is not so much to dictate an inner architecture for implementations. It is more to provide a way to strictly define the external behaviour of implementations. The specification defines the required behaviour of any Java virtual machine implementation regarding these abstract components and their interactions.

Let's discuss what does JVM performs?

It is used to load the code.

It Verifies and executes the code.

And it provides run time environment.

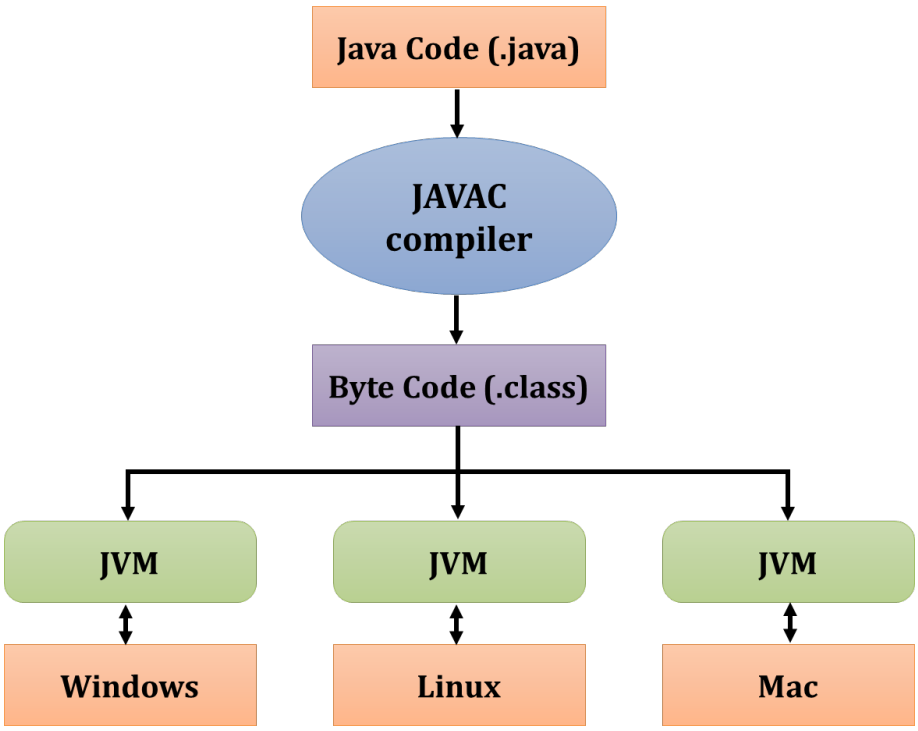


Figure 1.1.2: Java Virtual Machine

Software requirements for Java:

To run other users' Java programs, a user must have the JRE (Java Runtime Environment). The user must have the JDK (Java Development Kit) to write and run the Java programs, which is sometimes called the System Development Kit (SDK).Do not get both JDK and JRE because JDK includes JRE and both are the same.

The user also needs the largest and greatest version of Java which is known as "Java 2 Platform, Standard Edition 5.0."

The user needs a text editor with which to write programs. Recommended editors are Text Pad on Windows and BBEdit on the Macintosh. However, most IDEs include a text editor.

An Integrated Development Environment (IDE) is a single program that lets a user to edit, compile, run, test and debugs programs, all in one place. If a user does not have an IDE, he needs to learn how to compile and run Java programs from the command line which is not a difficult task, just less convenient.

There are various softwares for users:

- **JRE (Java Runtime Environment):** The JRE provides the libraries, the java virtual machine and components to run applets and application written in the Java programming language.
- **JDK (Java Development Kit):** JDK is the official development kit for the Java programming language. JDK contains the software and tools that user needs to compile, debug and run applets and applications written using the java programming language.
- **Eclipse:** It is an excellent and very popular IDE. Eclipse is a software designed to manage IDE for Java language. It is helpful for programming applications for Window platforms.
- **NetBeans:** It is Sun's IDE and catching up with Eclipse. The main advantage of NetBeans is that it can be downloaded along with the JDK. It also provides a visual GUI builder, but that relies on some packages that must be included with the program if it is to run outside the NetBeans environment.

How to set path to run a program?

The path is required to be set for using tools such as javac, java etc. If user is saving the java source file inside the jdk/bin directory, path is not required to be set because all the tools will be available in the current directory. But if user having his java file outside the jdk/bin folder, it is necessary to set path of JDK.

There are two ways to set java path:

1. Temporary
2. Permanent

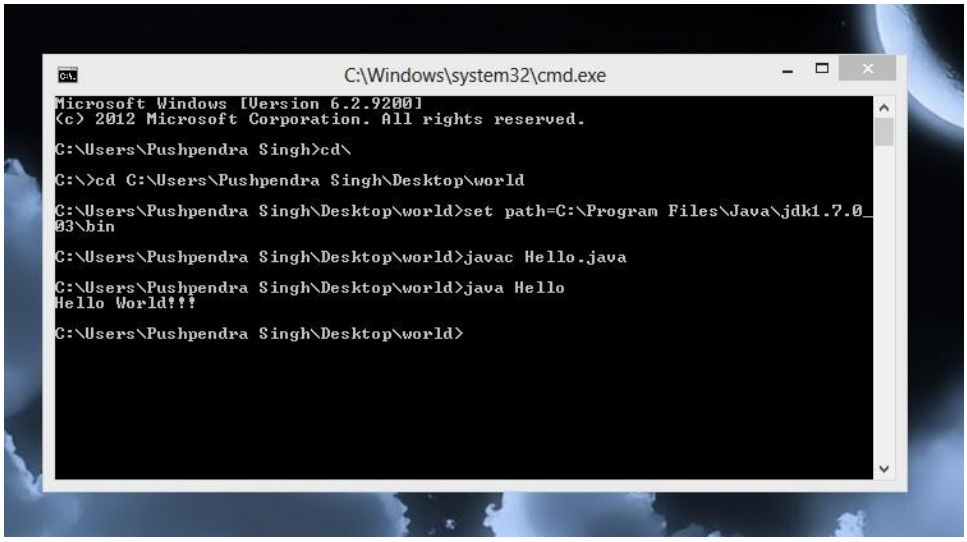
How to set Temporary Path of JDK in windows?

To set the temporary path of JDK, user needs to follow following steps:

1. Open command prompt
2. Copy the path of jdk/bin directory
3. Write in command prompt: setpath=copied_path

For example:

set path=C:\Program Files\Java\jdk1.7.0_03\bin



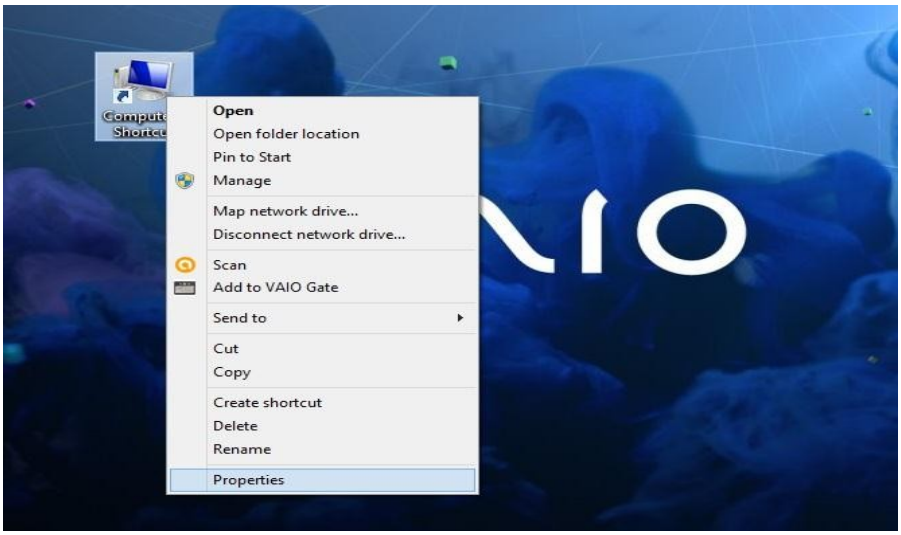
How to set Permanent Path of JDK in windows?

For setting the permanent path of JDK, user needs to follow these steps:

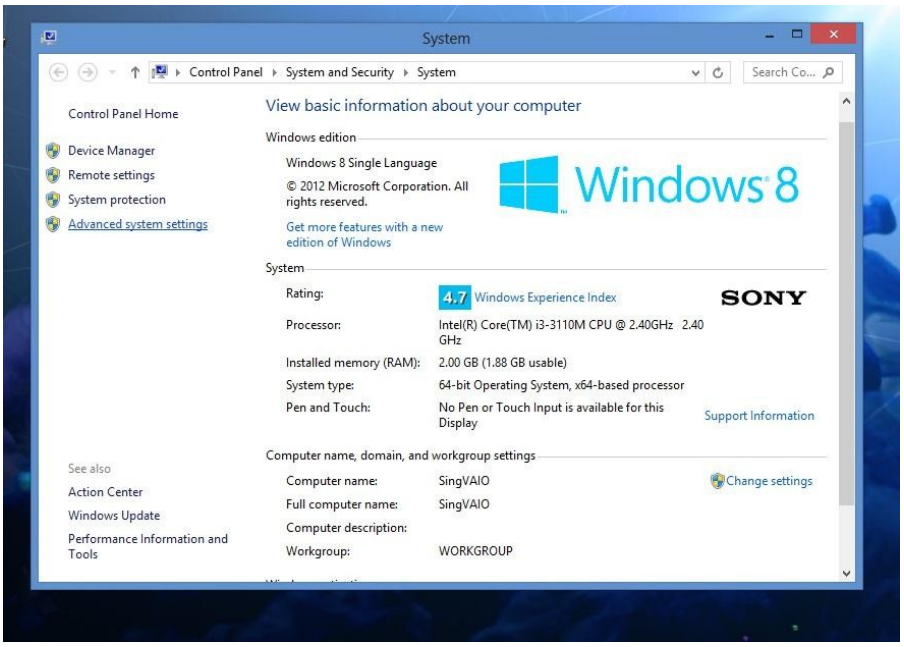
Go to My Computer properties -> advanced tab -> environment variables ->new tab of user variable -> write path in variable name ->write path of bin folder in variable value -> Click ok-> ok->ok

For Example:

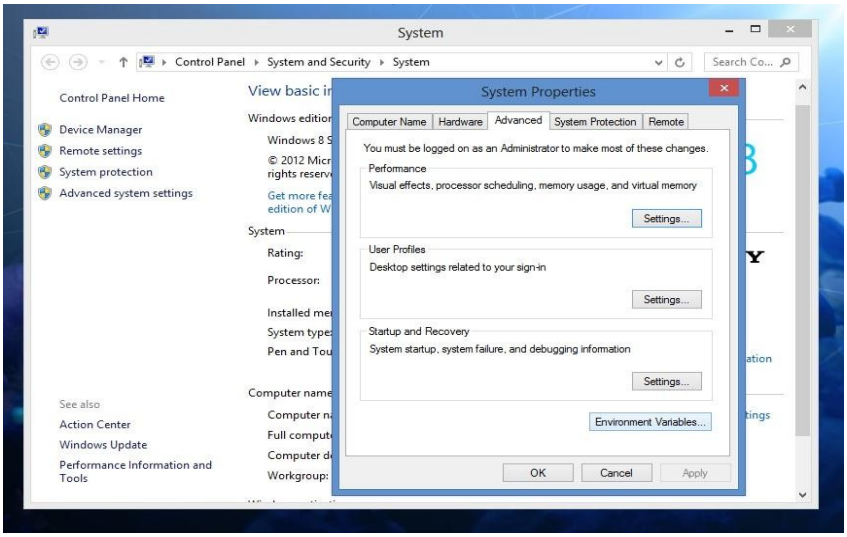
Step 1: Go to My computer properties



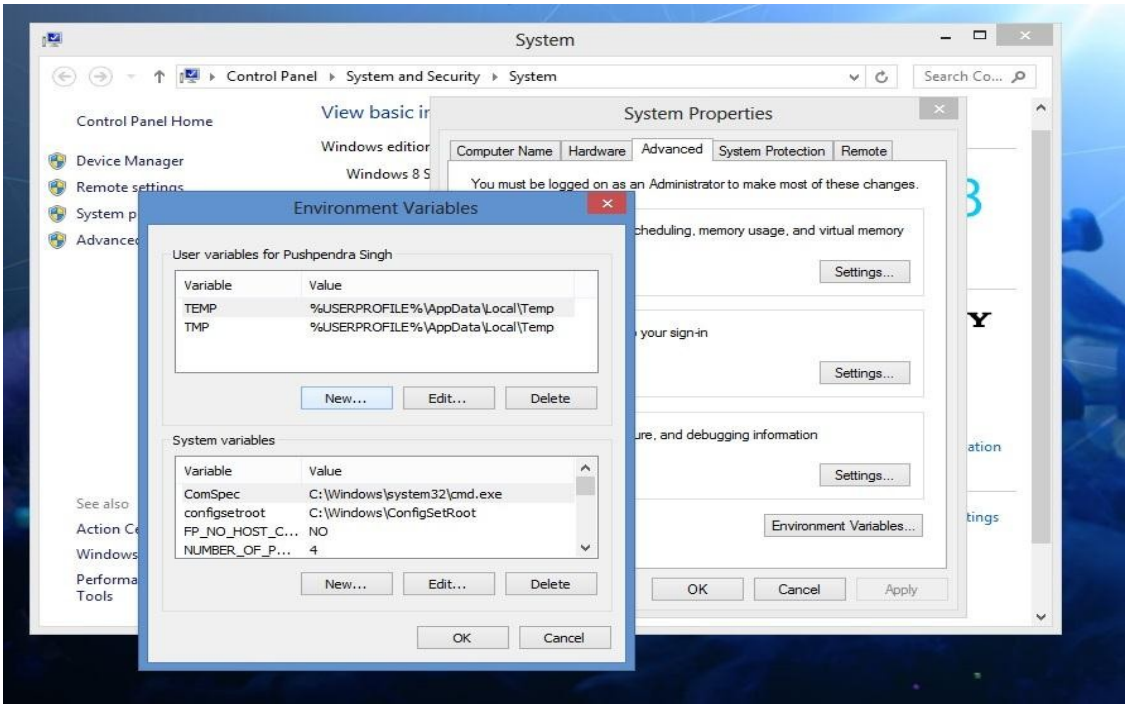
Step 2: Click on the advance tab



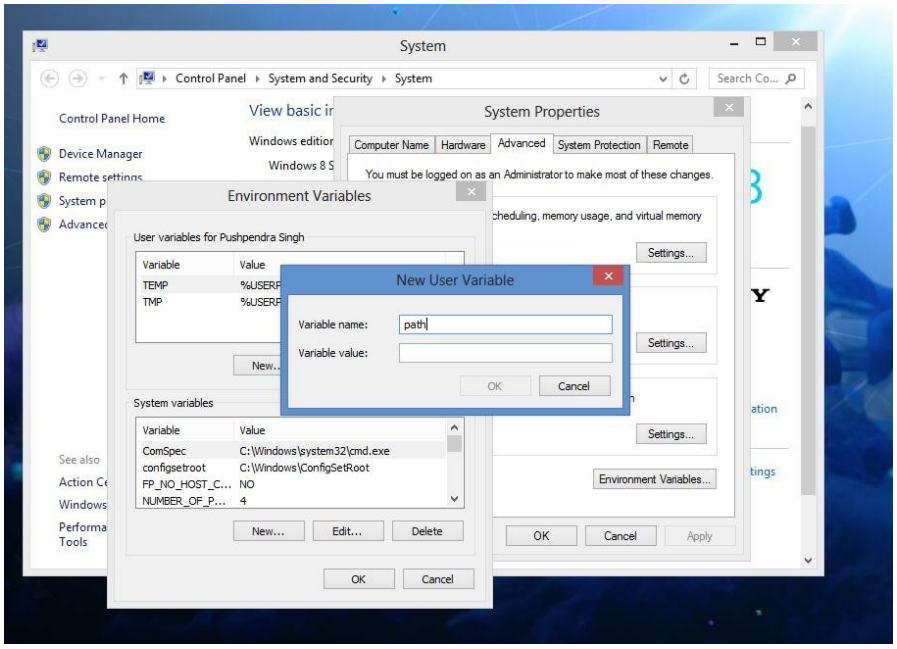
Step 3: Click on the environment variables



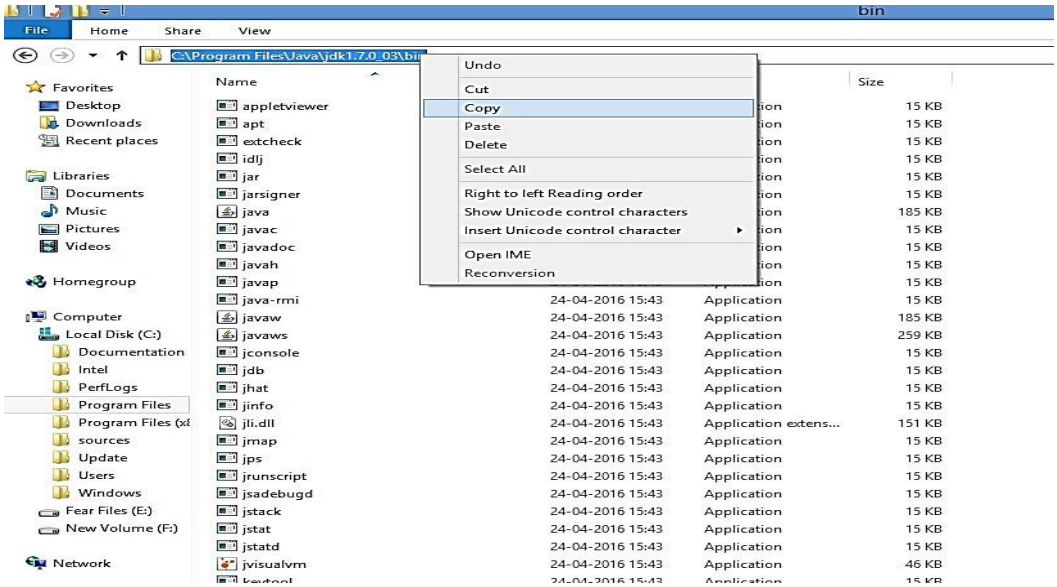
Step 4: Click on new tab of user variables



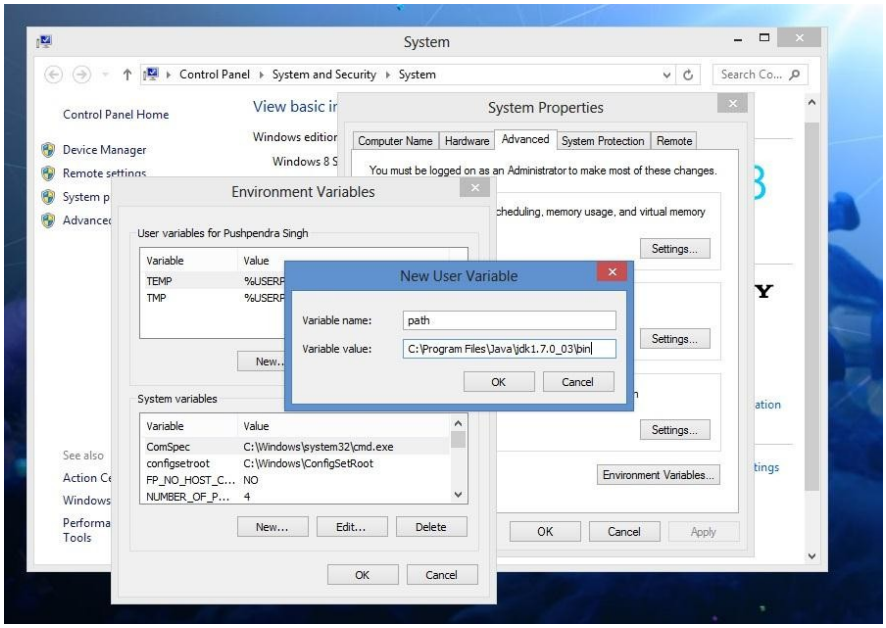
Step 5: Write path in variable name



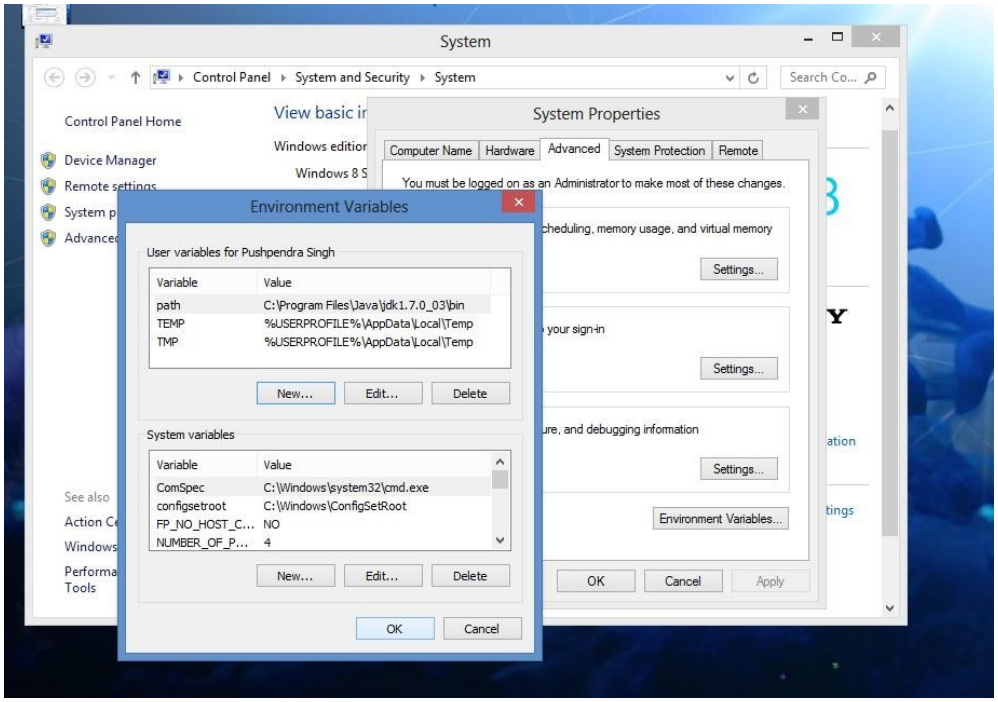
Step 6: Copy the path of bin folder



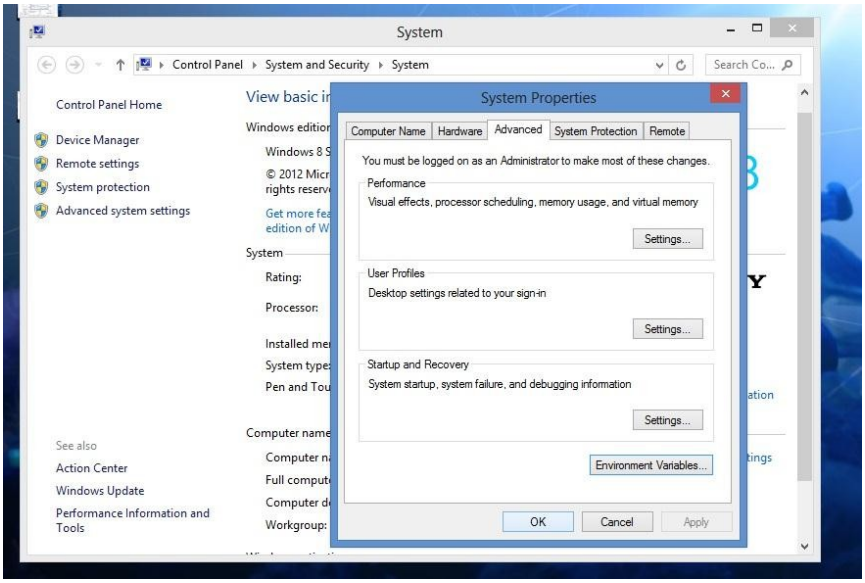
Step 7: Paste path of bin folder in variable value



Step 8: Click on OK Button



Step 9: Click on OK button



Now the permanent path is set. User can execute any program of java from any drive

Simple Java Program:

Java is a general-purpose programming language. Like any programming language, Java also has its syntax, structure and programming paradigm to build a robust, reusable and maintainable applications programs.

Java programs are executed in its development environments such as Java Development Kit (JDK) and Eclipse IDE.

Programmatically, Java is a derivative of C++language. The Syntax of Java looks similar like C. Code blocks are modularised into methods and delimited by braces ({ }) and variables are declared before they are used.

Java program starts with a package where the package can be defined as a namespace mechanism. It helps to maintain the hierarchical file system, manage file system and class system.

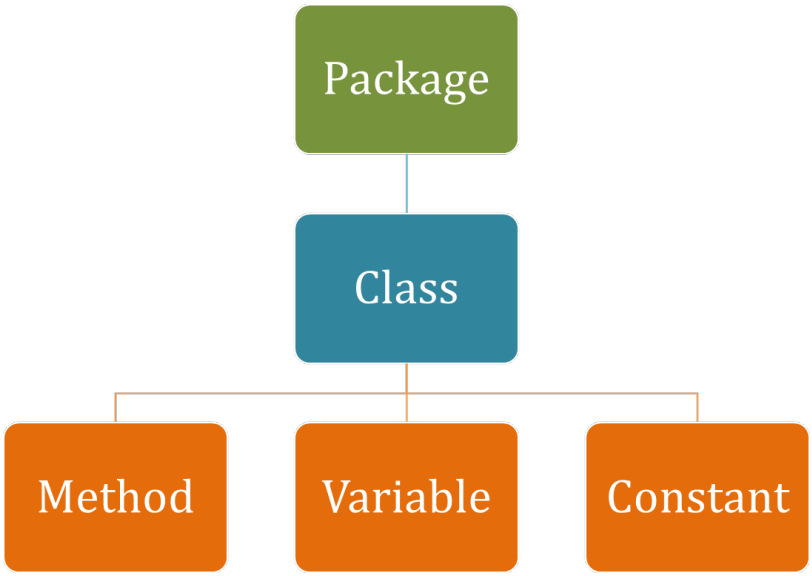


Figure 1.1.5: Hierarchy of Java Program Components

A simple Java program is given below:

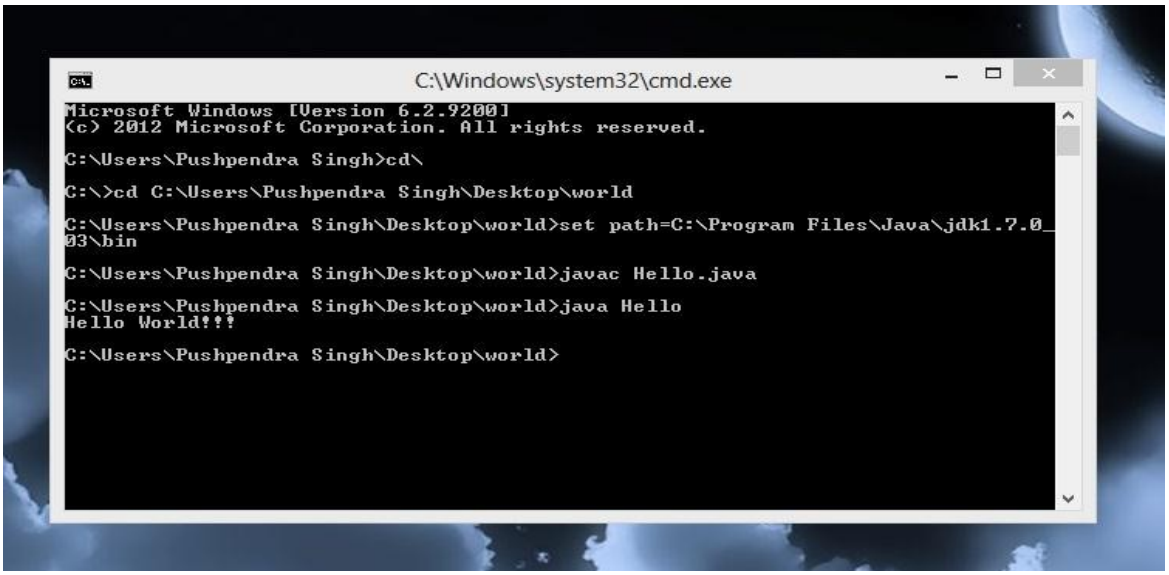
Example program

```
/*
*First java program
*/
class Hello //save as Hello.java {
Public static void main (string args[]) // program entry point {
System.out.println ("Hello world!!!"); // print message
```

} //end of main

} //end of First Program class

Output of this program: Hello world!!!



Brief explanation of this above sample example:

/* ..... */

// ... until the end of the line

/*---*/ It means comments. Comments are un-executable and are ignored by the compiler. But they provide useful explanation and documentation to your readers. There are two kinds of comments:

1. Multi-Line Comment: begins with /* and ends with */, and may span more than one lines.

2. End-of-Line (Single-Line) Comment: begins with // and lasts until the end of the modern line.

```
classFirstClass {
   //
   //
}
```

It is a class declaration. The area between the braces called body which contains all the code that provides for the life cycle of the objects created from the class.

The above class declaration is a minimal one. It contains only those components of a class declaration that are required. The programmer can provide many more information about the class.

For example,

```
classFirstClass extends FirstSuperClass implements MyInterface {
   //
   //
}
```

It means that FirstClass is a subclass of FirstSuperClass and that it implements the MyInterface interface.

Generally, class declarations can include these components, in order:

- Modifiers such as public, private, and some others that you will encounter later.
- The class name, with the initial letter capitalised by convention.
- The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
- A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
- The class body, surrounded by braces, {}.

public static void main(String[] args)

{.....

Let's discuss about public, static, void and main ():

- **public:** public means, public so that everyone can access this method.
- **static:** Static means that this method can be called without instantiating the class.
- **void**: void means this method does not return anything or main() returns no value.
- **main():** It is a method or a function name which is configured to JVM (Java virtual machine)

- **(String [] args):** It means that arguments to the main method i.e. Command line arguments
- The braces {.....} encloses the body of the method, which contains programming statements.

System.out.println("Hello, world!");

Let's discuss about System, out and println ():

- **System:** It is a class name which is present in java.lang
- **out:** out means that thestatic variable of type print stream present in system class
- **println():** It is a method which is present in print stream class

The programming statement System.out.println ("Hello, world!") is used to print the string "Hello, world!" to the display console. A string is surrounded by a pair of double quotes and contain texts. The text will be printed as it is, without the double quotes. A programming statement ends with a semi-colon (;).

Data Types, Keyword and Variables

Variables are memory locations to store values. When a user creates a variable, a memory space is reserved for that the variable. Based on the type of the variable, the operating system allocates memory and decides what can be stored in that reserved space. A user can store integers, decimals, character, etc. by assigning different data types to the variables. Data type is a keyword used to allocate sufficient memory space for the data. Pictorial representation of data types and its classifications are given below:

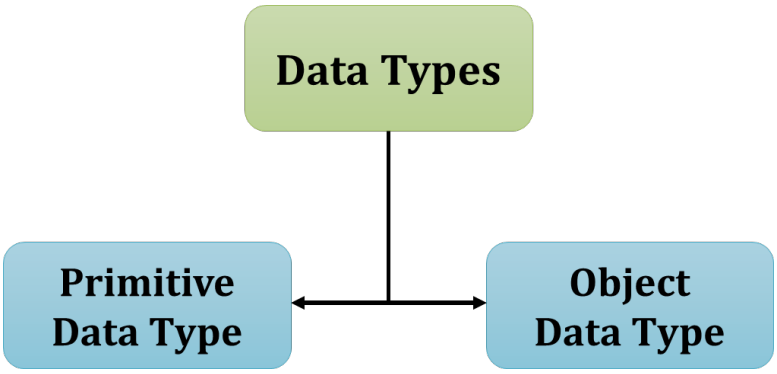


Figure 1.1.6: Classification of Data Types

Let's see the list of data types and its description in the following. The types are:

Integers

Java defines **four** integer types:

1. byte,
2. short,
3. int and
4. long.

All of these are signed, positive and negative values. Java does not support unsigned data types. Whereas signed and unsigned integers are supported by many other computer languages. Java designers felt that unsigned integers are unnecessary because they are mostly used to specify behaviour of the higher order bit, which defines the sign of an integer value. The width of an integer type should not be thought of as the amount of storage it consumes, but rather as the behaviour it defines for variables and expressions of that type. The Java run-time environment is free to use whatever size it wants, as long as the types behave as you declared them. The width and ranges of these integer types vary widely, as shown in this table:

Name	Width	Range
long	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
int	32	–2,147,483,648 to 2,147,483,647
short	16	–32,768 to 32,768
byte	8	–128 to 127

Table 1.1.1: Integers and its Types

Byte

The smallest integer type is byte. This is a signed 8-bit type that has a range from -128 to 127. Variables of type byte are especially useful when you're working with a stream of data from a network or file. It can be represented as:

```
Byte b, c;
```

Short

Short is a signed 16-bit type. It has a range from –32,768 to 32,767. It is probably the least-used Java type.

Here are some examples of short variable declarations:

```
short s;
```

Int

The most commonly used integer type is int. It is a signed 32-bit type that has a range from –2,147,483,648 to 2,147,483,647.

In addition to other uses, variables of type int are commonly employed to control loops and to index arrays.

Long

Long is a signed 64-bit type and is useful for those occasions where an int type is not large enough to hold the desired value.

The range of a long is quite large. This makes it useful when big, whole numbers are needed.

Floating-point Types

Floating-point numbers, also known as real numbers

There are two kinds of floating-point types, float and double, which represent single - and double-precision numbers, respectively. Their width and ranges are shown here:

Name	Width in Bits	Approximate Range
double	64	4.9e-324 to 1.8e+308
float	32	1.4e-0.45 to 3.4e+0.38

Table 1.1.2: Floating-point Types

Float

Float is specified as a single-precision value that uses 32 bits of storage.

Single precision is faster and takes half as much space as double precision, but will become imprecise when the values are either very large or very small.

Here are some example float variable declarations:

float hightemp, lowtemp;

Double

Double is denoted by the keyword ‘double’ which is used to store a value of 64 bit. It is faster than single precision on modern processor that have been optimised for high-speed mathematical calculations. Some importance of double are:

All transcendental math functions, such as

- sin( )
- cos( )
- sqrt( )

Return double values.

Double is best choice, when you need to maintain accuracy over many iterative calculations, or are manipulating large-valued numbers.

Example Program for Integer:

```
Public class DataType_int
{
    int a=100; int
    b=-150; void
    add( )
    {
        Int c=a + b;
        Sysytem.out.println (“The int value is:” +c);
    }
}
Class Demo
{
    public static void main(String args[ ] )
    {
        Datatype_Int obj =new DataType_Int ( ); Obj.add();
    }
}
```

Output:

The int value is -50

Sample program for double variable:

```
// Compute the area of a circle. class Area
{
    public static void main(String args[])
    {
        double pi, r, a;
        r = 10.8; // radius of circle
        pi = 3.1416; // pi, approximately a = pi * r * r;
        // compute area
        System.out.println("Area of circle is " + a);
    }
}
```

Output:

Area of circle is 366.435

Characters

Char in Java is not the same as char in C or C++. Java uses Unicode to represent characters. Unicode defines a fully international character set that can represent all of the characters found in all human languages. It is a unification of dozens of character sets, such as

Latin

Greek

Arabic

Cyrillic

Hebrew

In Java, char is 16 bit and ranges from 0 to 65,536. There are no negative chars. The standard set of characters known as ASCII still ranges from 0 to 127 as always and the extended 8-bit character set, ISO-Latin-1, ranges from 0 to 255.

Here is a program that demonstrates char variables:

```
class CharDemo

{

    public static void main(String args[])

    {

        char ch1, ch2;

        ch1 = 88; // code for X ch2 = 'Y';

        System.out.println("ch1 and ch2: ");
        System.out.println(ch1 + " " + ch2);

    }

}
```

This program displays the following output:

ch1 and ch2: X Y

Boolean

The Boolean logical operators are:

| , & , ^ , ! , || , && , == , != .

Java supplies a primitive data type called Boolean, instances of which can take the value true or false only and have the default value false. The major use of Boolean facilities is to implement the expressions which control if decisions and while loops.

Sample Program:

```
class Boolean

{

    public static void main(String args[])

    {

        // these are boolean variables boolean A
        = true;

        boolean B = false;

        System.out.println ("A|B = "+(A|B));// Program execution using Boolean System.out.println
        ("A&B = "+(A&B));

        System.out.println ("!A = "+(!A)); System.out.println ("A^B =
        "+(A^B)); System.out.println ("(A|B)&A = "+((A|B)&A));

    }
```

}

Syntax:

booleanValue( ); // returns the value of Boolean object

Output:

```
A|B=true
A&B=false

!A=false
A^B=true
(A|B)&A=true
```

Type Conversion and casting:

Type casting is used to assign a value of one type of a variable to another type. Casting in other words is an instruction to the compiler to convert one type to another. It helps Java not only to perform only arithmetic operations on pairs of values of the same type but performs operations on mixed type too.

For example, the value of the right side is automatically converted to the type of the left side in compatible types. In Java, Boolean and int are not compatible.

int x=10;

byte y= (byte)x;

When the above said conditions are met, a widening conversion takes place. In Java type casting are classified as two types

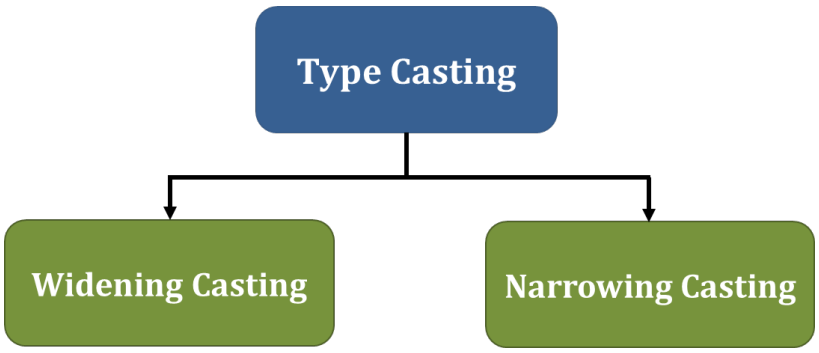
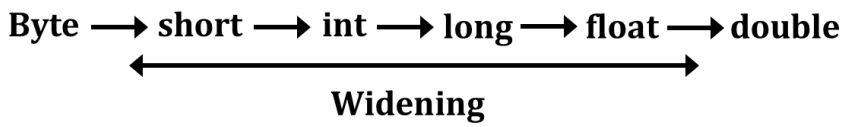


Figure 1.1.7: Type Casting and its Classification

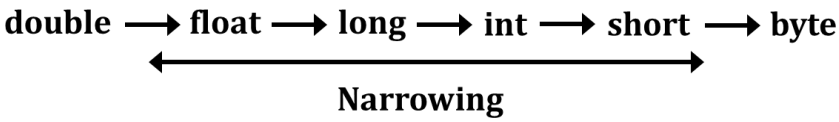
Widening casting

In widening casting, the value types are not directly expressed. Here, integer and floating are compatible with each other. An Integer literal can be assigned to char.



Narrow casting:

In narrow casting, the value types are directly expressed.



Let's see the difference between widening casting and narrow casting for finer understanding.

Widening Casting	Narrow Casting
Used to access new methods of child class or while referring super class to child class	Used to access redefined class of parent class in the child class

Table 1.1.3: Difference between Widening and Narrowing Casting

Sample Program:

```
Package conversion;

Public class main

{

    Public Static Void main(string [ ] args)

    {

        int var1=12; double
        var2=15.2;

        double result = var1+var2; System.Out.Println ("The sum
        is" + result)

    }

}
```

Output: The sum is 27.2

Lexical Issues

Java programs are the collection of whitespace, identifiers, literals, comments and separators. Lexical issues are very common in programming, which is used to analysis, operate and implement the program in a sequential order.

Let us see the detail description of lexical issues in the following.

Whitespace

As word by itself, a whitespace is nothing but space, tab, or a new line. Java is a case sensitive free-form language which means a user need not follow any special rules. ***For example,*** there is no format to be followed to write a Java program. A program can be written even in a single line or any number of lines can be used, whatever the user wishes. By default, single whitespace is required between each character.

Identifiers

An identifier can be described as any descriptive sequence of uppercase, lowercase letters, underscore or a dollar - sign character. Identifiers are used for specifying the class name, method name and variable name. Since Java is case-sensitive language any letter of a word will be different from the other form of the similar word. Let us say in Java, a word max is different from MAX.

Literals

Literals are used to specify a default value or to declare a constant value.

It can be a:

Boolean – True or False

Numeric – Floating, decimal, integer or a fixed point

Character based – Single character or a string

Comments

Three types of comments are supported by Java. They are:

Comment	Description
/*text*/	Compiler ignores everything from /* to */

//text	Compiler ignores everything from // to the end of the line
//documentation*/**	Called as doc comment (documentation comment) – used by automatically generated document

Table 1.1.4: Comments and its Description

Separators

Separators are characters used to terminate statements. Commonly used separators are the semicolon (;). Following are few separators used in Java programs

Symbol	Name	Purpose
( )	Parenthesis	Control the order of operation
{ }	Braces	Used to group statements and declaration
[ ]	Brackets	Used to declare array types and dereferencing array values
;	semicolon	Terminates statement
,	comma	Separates consecutive identifiers in a variable declaration
.	period	Used to separate package name from sub packages and classes

Table 1.1.5: Separators and its Purpose

Arrays

The Array is a collection of similar types of elements that have contiguous memory location. It is a container object that holds values of a homogeneous type.

Java array is container object that contains elements of similar data type. It can have:

Only fixed set of elements which can be stored in it.

Also, have label as static data structure because during the time of declaration only the size of the array is specified.

An array containing a number of variables and it can even start from zero – which is called as an empty. Instead they are referenced by array access expressions that use non-negative integer index values.

Syntax:

datatype[ ] identifier;

Or

datatype identifier;

Initialisation of an array:

[ ] - Int operator is used to initialise an array.

For example, program

Int [ ] arr = new int [15]; // 15 is the size of an array.

Array Type:

The Array types are used in declarations and in cast expression. An array type is written as the name of an element type followed by some number of empty pairs of square brackets [ ]. The number of bracket pairs indicates the depth of array nesting.

An array's length is not part of its type. It may be of any type, whether primitive or reference.

Arrays with an abstract type as their element type are allowed. Elements of such an array may have as their value null or a reference to an instance of any type that implements the abstract type.

An array may have as its value a reference to an instance of any subclass of an abstract class that is not itself abstract.

Array Variable:

A variable of an array type enfolds a reference to an object. It is used to create a variable and does not create any space to that specified variable. Some of the variable declarations are listed below. They are:

Declaration	comment
Int [ ]	Array of int
Short [ ] [ ]	Array of short
Object [ ]	Array of object
Collection<?>	Array of collection of unknown types

Table 1.1.6: Array Variable

Single Dimensional Array:

A one-dimensional array consist of a list of like-typed variables.

The general form of a one-dimensional array declaration is

```
type var-name[];
// syntax
```

The general form of new as it applies to one-dimensional arrays, appears as follows:

arrayRefVar =new dataType[arraySize];

The above statement does two things:

It creates an array using new dataType[arraySize];

It assigns the reference of the newly created array to the variable arrayRefVar.

The example below allocates a 12-element array of integers and links them to month_days.

month_days = new int[12];

After this statement executes, month_days will refer to an array of 12 integers. Further, all elements in the array will be initialised to zero.

```
// Demonstrate a one-dimensional array.
class Array

{

    public static void main(String args[])

    {

        int month_days[];
        month_days = new int[12];
        month_days [0] = 31;

        month_days [1] = 28;

        month_days [2] = 31;

        month_days [3] = 30;

        month_days [4] = 31;

        month_days [5] = 30;

        month_days [6] = 31;

        month_days [7] = 31;

        month_days [8] = 30;

        month_days [9] = 31;

        month_days [10] = 30;

        month_days [11] = 31;

        System.out.println ("April has " + month_days[3] + " days.");

    }

}
```

This program generates the following output:

April has 30 days

When you run this program, it prints the number of days in April. As mentioned, Java array indexes start with zero, so the number of days in April is month_days[3] or 30.

It is possible to combine the declaration of the array variable with the allocation of the array itself, as shown here:

```
int month_days[ ] = new int[12];
```

Multidimensional Array

Java does not support multidimensional arrays. However, you can declare and create an array of arrays

However, as you will see, there are a couple of subtle differences. To declare a multidimensional array variable, specify each additional index using another set of square brackets. ***For example,*** the following declares a two dimensional array variable called twoD.

```
int twoD [ ] [ ] = new int[4][5];
```

The following program numbers each element in the array from left to right, top to bottom and then displays these values:

```
// Demonstrate a two-dimensional array.
class TwoDArray


{

    public static void main(String args[])

    {

        int twoD[ ] [ ]= new int[4][5]; int i, j,
        k = 0;

        for(i=0; i<4; i++)
        for(j=0; j<5; j++)

        {

            twoD[i][j] =
            k; k++;

        }

        for(i=0; i<4; i++)

        {

            for(j=0; j<5; j++)
            System.out.print(twoD[i][j] + " ");
            System.out.println();

        }

    }
```

```
    }
```

This program generates the following output:

```
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
```