Introduction to java

1. History of Java:

- In 1990, Sun Micro Systems Inc. (US) was conceived a project to develop software for consumer electronic devices that could be controlled by a remote. This project was called Stealth Project but later its name was changed to Green Project.
- In January 1991, Project Manager James Gosling and his team members Patrick Naughton, Mike Sheridan, Chris Wrath, and Ed Frank met to discuss about this project.
- Gosling thought C and C++ would be used to develop the project. But the problem he faced with them is that they were system dependent languages. The trouble with C and C++ (and most other languages) is that they are designed to be compiled for a specific target and could
  not be used on various processors, which the electronic devices might use.
- James Gosling with his team started developing a new language, which was completely system independent. This language was initially called **OAK**. Since this name was registered by some other company, later it was changed to **Java**.
- James Gosling and his team members were consuming a lot of coffee while developing this language. Good quality of coffee was supplied from a place called "Java Island'. Hence they fixed the name of the language as Java. The symbol for Java language is cup and saucer.· Sun formally announced Java at Sun World conference in 1995. On January 23rd 1996,
  JDK1.0 version was released

Features of Java (Java buzz words):

Simple: Learning and practicing java is easy because of resemblance with c and C++.

Object Oriented Programming Language: Unlike C++, Java is purely OOP.

Distributed: Java is designed for use on network; it has an extensive library which works in agreement with TCP/IP.

Secure: Java is designed for use on Internet. Java enables the construction of virus-free, tamper free systems.

Robust (Strong/ Powerful): Java programs will not crash because of its exception handling and its memory management features.

Interpreted: Java programs are compiled to generate the byte code. This byte code can be downloaded and interpreted by the interpreter. .class file will have byte code instructions and JVM which contains an interpreter will execute the byte code.

Portable: Java does not have implementation dependent aspects and it yields or gives same result on any machine.

Architectural Neutral Language: Java byte code is not machine dependent, it can run on any machine with any processor and with any OS.

High Performance: Along with interpreter there will be JIT (Just In Time) compiler which enhances the speed of execution.

Multithreaded: Executing different parts of program simultaneously is called multithreading. This is an essential feature to design server side programs.

Dynamic: We can develop programs in Java which dynamically change on Internet (e.g.:Applets).

Obtaining the Java Environment:

We can download the JDK (Java Development Kit) including the compiler and runtime engine from Sun at: http://java.sun.com/javase.

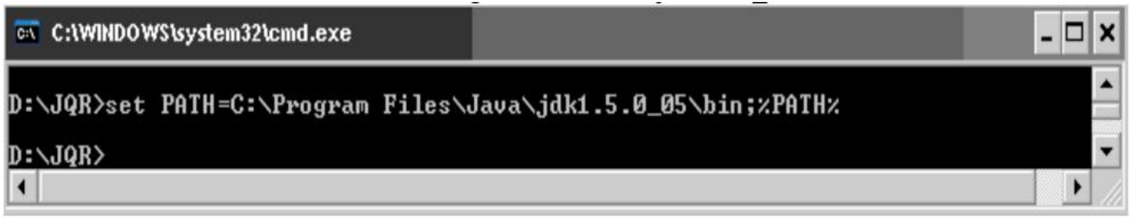
Install JDK after downloading, by default JDK will be installed in

C:\Program Files\Java\jdk1.5.0_05 (Here jdk1.5.0_05 is JDK's version)

Setting up Java Environment: After installing the JDK, we need to set at least one environment variable in order to able to compile and run Java programs. A PATH environment variable enables the operating system to find the JDK executables when our working directory is not the JDK's binary directory.

Setting environment variables from a command prompt: If we set the variables from a command prompt, they will only hold for that session. To set the PATH from a command prompt:

set PATH=C:\Program Files\Java\jdk1.5.0_05\bin;%PATH%



```
D:\JQR>set PATH=C:\Program Files\Java\jdk1.5.0_05\bin;%PATH%
D:\JQR>
```

2. Programming Structure:

Comments: Comments are description about the aim and features of the program. Comments increase readability of a program.

Three types of comments are there in Java:

- **Single line comments:** These comments start with //

  e.g.: // this is comment line

- **Multi line comments:** These comments start with /* and end with */

  e.g.: /* this is comment line*/

- **Java documentation comments:** These comments start with /** and end with */

These comments are useful to create a HTML file called API (application programming Interface) document. This file contains description of all the features of software.

Structure of the Java Program:

- As all other programming languages, Java also has a structure.
- The first line of the C/C++ program contains include statement. For example, <stdio.h> is the header file that contains functions, like printf (), scanf () etc. So if we want to use any of these functions, we should include this header file in C/ C++ program.

- Similarly in Java first we need to import the required packages. By default java.lang.* is imported. Java has several such packages in its library. A package is a kind of directory that contains a group of related classes and interfaces. A class or interface contains methods.
- Since Java is purely an Object Oriented Programming language, we cannot write a Java program without having at least one class or object. So, it is mandatory to write a class in Java program. We should use class keyword for this purpose and then write class name.
- In C/C++, program starts executing from main method similarly in Java, program starts executing from main method. The return type of main method is void because program starts executing from main method and it returns nothing.

Sample Program:

```
//A Simple Java Program
class Sample
{
public static void main(String args[])
{
System.out.print ("Hello world");
}
}
```

- Since Java is purely an Object Oriented Programming language, without creating an object to a class it is not possible to access methods and members of a class. But main method is also a method inside a class, since program execution starts from main method we need to call main method without creating an object.
- Static methods are the methods, which can be called and executed without creating objects.
- Since we want to call main () method without using an object, we should declare main () method as static. JVM calls main () method using its Class name.main () at the time of running the program.

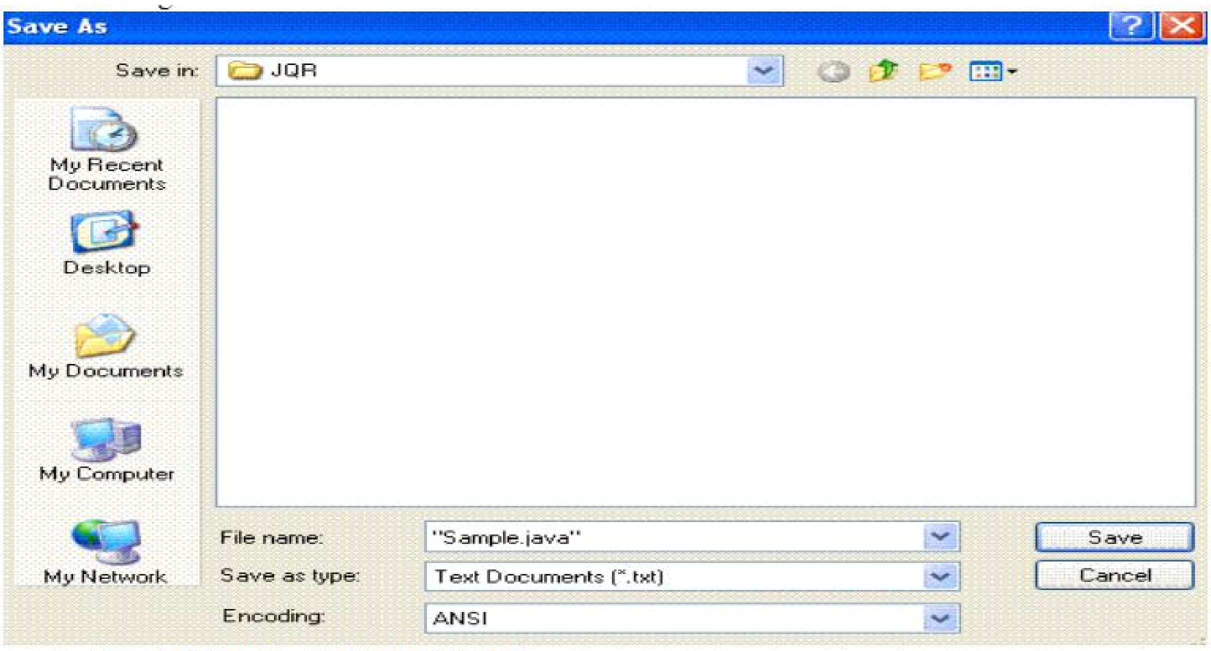
- JVM is a program written by Java Soft people (Java development team) and main () is the method written by us. Since, main () method should be available to the JVM, it should be declared as public. If we don't declare main () method as public, then it doesn't make itself available to JVM and JVM cannot execute it.
- JVM always looks for main () method with String type array as parameter otherwise JVM cannot recognize the main () method, so we must provide String type array as parameter to main () method.
- A class code starts with a { and ends with a }. A class or an object contains variables and methods (functions). We can create any number of variables and methods inside the class.

  This is our first program, so we had written only one method called main ().
- Our aim of writing this program is just to display a string "Hello world". In Java, print () method is used to display something on the monitor.

Creating a Source File:

- Type the program in a text editor (i.e. Notepad, WordPad, Microsoft Word or Edit Plus). for select a notepad editor from your computer .
- In a new document, type the above code (i.e. Sample Program).
- Save the program with filename same as Class_name (i.e. Sample.java) in which main method is written. To do this in Notepad, first choose the **File > Save** menu item. Then, in the **Save** dialog box:
- Using the **Save in** combo box, specify the folder (directory) where you'll save your file.

  In this example, the directory is JQR on the D drive.
- In the **File name** text field, type "Sample.java", including the quotation marks. Then the dialog box should look like this:

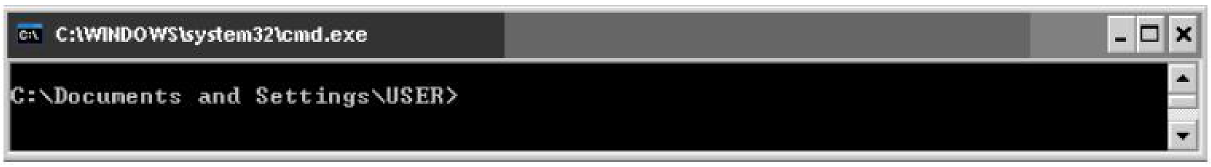


File name: "Sample.java"
Save as type: Text Documents (*.txt)
Encoding: ANSI

- Now click **Save**, and exit Notepad.

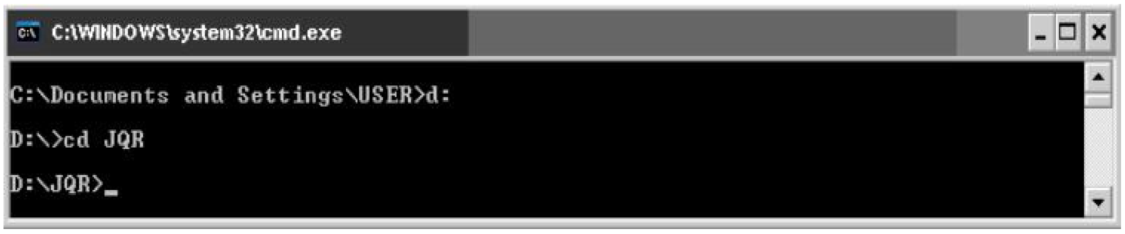
Compiling the Source File into a .class File:

- To Compile the Sample.java program go to DOS prompt. We can do this from the **Start** menu by choosing **Run...** and then entering cmd. The window should look similar to the following figure.



```
C:\Documents and Settings\USER>
```

- The prompt shows current directory. To compile Sample.java source file, change current directory to the directory where Sample.java file is located. For example, if source directory is JQR on the D drive, type the following commands at the prompt and press Enter:

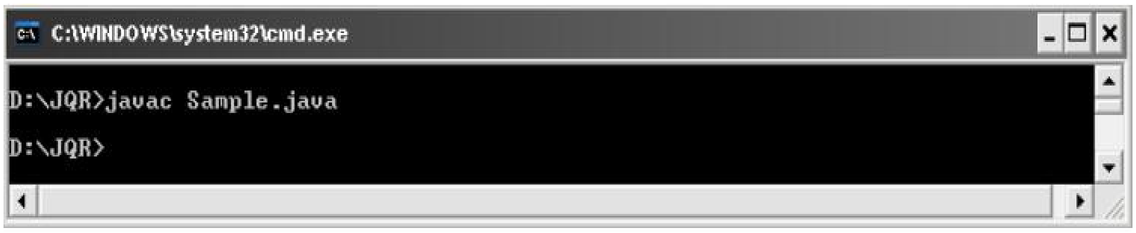


```
C:\Documents and Settings\USER>d:
D:\>cd JQR
D:\JQR>_
```

Now the prompt should change to D:\JQR>

- At the prompt, type the following command and press Enter.

  javac Sample.java



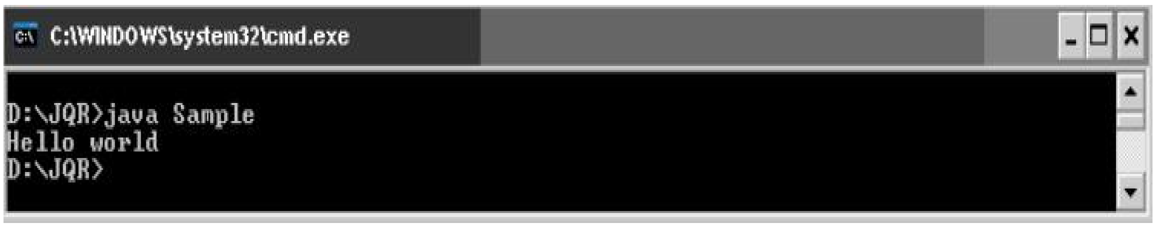
C:\WINDOWS\system32\cmd.exe

```
D:\JQR>javac Sample.java

D:\JQR>
```

· The compiler generates byte code and Sample.class will be created

Executing the Program (Sample.class):

➢ To run the program, enter java followed by the class name created at the time of compilation at the command prompt in the same directory as:
java Sample



C:\WINDOWS\system32\cmd.exe

```
D:\JQR>java Sample
Hello world
D:\JQR>
```

➢ The program interpreted and the output is displayed

What we are discussing until now is the java structure and the way we can run compile java program by using cmd after we would created a system variable on our computer. However we compile java programs by using GUI compilers like NetBeans and Eclips.

Naming Conventions, Data Types and Operators

Naming Conventions: Naming conventions specify the rules to be followed by a Java programmer while writing the names of packages, classes, methods etc.

Package names are written in small letters.

e.g.: java.io, java.lang, java.awt etc

· Each word of class name and interface name starts with a capital

e.g.: Sample, AddTwoNumbers

· Method names start with small letters then each word start with a capital

e.g.: sum (), sumTwoNumbers (), minValue ()

· Variable names also follow the same above method rule

e.g.: sum, count, totalCount

· Constants should be written using all capital letters

e.g.: PI, COUNT

· Keywords are reserved words and are written in small letters.

e.g.: int, short, float, public, void

Data Types: The classification of data item is called data type. Java defines eight simple types
of data. byte, short, int, long, char, float, double and boolean. These can be put in four groups:

✓ **Integer Data Types:** These data types store integer numbers

Data Type	Memory size	Range
Byte	1 byte	-128 to 127
Short	2 bytes	-32768 to 32767
Int	4 bytes	-2147483648 to 2147483647
Long	8 bytes	-9223372036854775808 to 9223372036854775807

✓ **Float Data Types:** These data types handle floating point numbers

Data Type	Memory size	Range
Float	4 bytes	-3.4e38 to 3.4e38
Double	8 bytes	-1.7e308 to 1.7e308

✓ **Character Data Type:** This data type represents a single character. char data type in java uses two bytes of memory also called Unicode system. Unicode is a specification to include alphabets of all international languages into the character set of java

Data Type	Memory size	Range
Char	2 bytes	0 to 65535

e.g.: char ch = 'x';

✓ **Boolean Data Type:** can handle truth values either true or false

e.g.:- boolean response = true;

■ **Operators:** An operator is a symbol that performs an operation. An operator acts on variables called operands.

● **Arithmetic operators:** These operators are used to perform fundamental operations like addition, subtraction, multiplication etc

Operator	Meaning	Example	Result
+	Addition	3 + 4	7
-	Subtraction	5 - 7	-2
*	Multiplication	5 * 5	25
/	Division (gives quotient)	14 / 7	2
%	Modulus (gives remainder)	20 % 7	6

- **Assignment operator:** This operator (=) is used to store some value into a variable.

Simple Assignment	Compound Assignment
x = x + y	x += y
x = x – y	x -= y
x = x * y	x *= y
x = x y	x /= y

- **Unary operators:** As the name indicates unary operator's act only on one operand.

Operator	Meaning	Example	Explanation
-	Unary minus	j = -k;	k value is negated and stored into j

++	Increment Operator	b++;
		++b;
			b value will be incremented by 1 (called as post incrementation)
			b value will be incremented by 1 (called as pre incrementation)

--	Decrement Operator	b--; --b;	b value will be decremented by 1 (called as post decrementation) b value will be decremented by 1 (called as pre decrementation)

- **Relational operators:** These operators are used for comparison purpose.

Operator	Meaning	Example
==	Equal	x == 3
!=	Not equal	x != 3
<	Less than	x < 3
>	Greater than	x > 3
<=	Less than or equal to	x <= 3

- **Logical operators:** Logical operators are used to construct compound conditions. A compound condition is a combination of several simple conditions.

Operator	Meaning	Example	Explanation

&&	and operator	if(a>b && a>c)
		System.out.print("yes");
			If a value is greater than b and c then only yes is displayed

||	or operator	if(a==1 || b==1)
		System.out.print("yes");
			If either a value is 1 or b value is 1 then yes is displayed

!	not operator	if(!(a==0))
		System.out.print("yes");
			If a value is not equal to zero then only yes is displayed

- **Bitwise operators:** These operators act on individual bits (0 and 1) of the operands. They act only on integer data types, i.e. byte, short, long and int.

Operator	Meaning	Explanation
&	Bitwise AND	Multiplies the individual bits of operands
|	Bitwise OR	Adds the individual bits of operands
^	Bitwise XOR	Performs Exclusive OR operation
<<	Left shift	Shifts the bits of the number towards left a specified number of positions
>>	Right shift	Shifts the bits of the number towards right a specified number of positions and also preserves the sign bit.
>>>	Zero fill right shift	Shifts the bits of the number towards right a specified number of positions and it stores 0 (Zero) in the sign bit.
~	Bitwise complement	Gives the complement form of a given number by changing 0's as 1's and vice versa.

- **Ternary Operator or Conditional Operator (? :):** This operator is called ternary because it acts on 3 variables. The syntax for this operator is:

  Variable = Expression1? Expression2: Expression3;

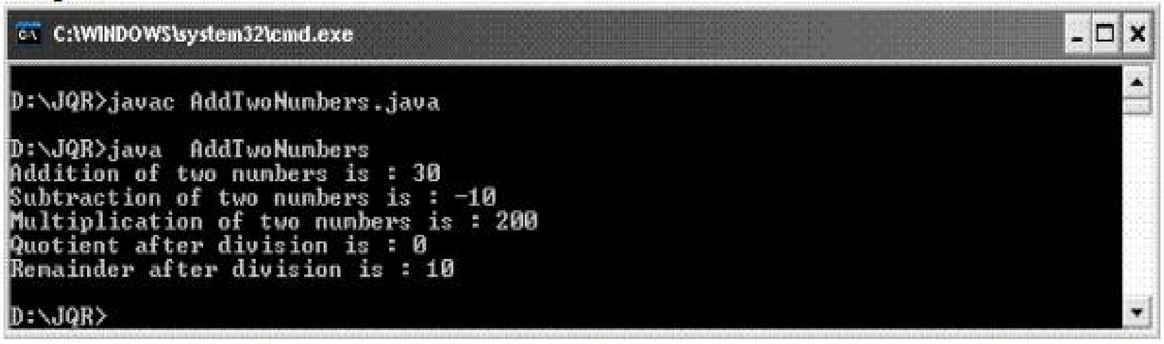
First Expression1 is evaluated. If it is true, then Expression2 value is stored into variable otherwise Expression3 value is stored into the variable.

e.g.: max = (a>b) ? a: b;

Program 1: Write a program to perform arithmetic operations

```
//Addition of two numbers
class AddTwoNumbers
{               public static void mian(String args[])
                {       int i=10, j=20;
                        System.out.println("Addition of two numbers is : " + (i+j));
                        System.out.println("Subtraction of two numbers is : " + (i-j));
                        System.out.println("Multiplication of two numbers is : " + (i*j));
                        System.out.println("Quotient after division is : " + (i/j) );
                        System.out.println("Remainder after division is : " +(i%j) );
                }
}
```

Output:

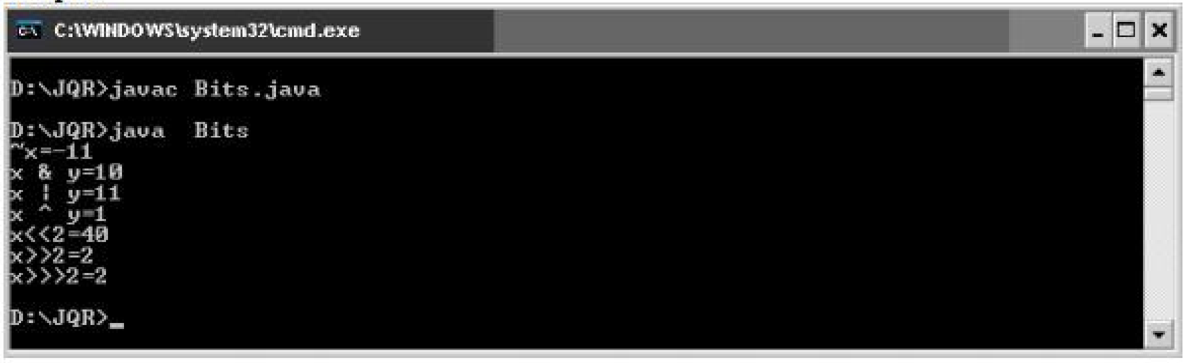


```
D:\JQR>javac AddTwoNumbers.java

D:\JQR>java  AddTwoNumbers
Addition of two numbers is : 30
Subtraction of two numbers is : -10
Multiplication of two numbers is : 200
Quotient after division is : 0
Remainder after division is : 10

D:\JQR>
```

Program 2: Write a program to perform Bitwise operations

```
//Bitwise Operations
class Bits
{       public static void main(String args[])
        {       byte x,y;
                x=10;
                y=11;
                System.out.println ("~x="+(~x));
                System.out.println ("x & y="+(x&y));
                System.out.println ("x | y="+(x|y));
                System.out.println ("x ^ y="+(x^y));
                System.out.println ("x<<2="+(x<<2));
                System.out.println ("x>>2="+(x>>2));
                System.out.println ("x>>>2="+(x>>>2));
        }
}
```

Output:



C:\WINDOWS\system32\cmd.exe

```
D:\JQR>javac Bits.java

D:\JQR>java  Bits
~x=-11
x & y=10
x | y=11
x ^ y=1
x<<2=40
x>>2=2
x>>>2=2

D:\JQR>_
```