Chapter 1: Basic of Java:

1.1 Java History

Java is an object-oriented programming language developed by Sun Microsystems in 1991. Java was invented for the development of software for consumer electronic devices. The main aim had to make java simple, portable, reliable and more secured.

Year	Progress
1990	Sun decided to developed software that could be used for electronic devices
1991	The team was announcement of a new language named ―Oak
1993	The World Wide Web appeared on the Internet and transformed the text-based interface to a graphical environment.
1994	The team developed a new Web browser called ―Hot Java‖ to locate and run Applets.
1995	Oak was renamed to Java
1996	Java language is now famous for Internet programming as well as a general purpose OO language.
1997	Sun releases Java Development Kit(JDK 1.1)
1998	Sun releases Software Development Kit (SDK 1.2)
1999	Sun releases Java 2 platform Standard Edition (J2SE) and Enterprise Edition (J2EE).
2000	J2SE with SDK 1.3 was released.
2002	J2SE with SDK 1.4 was released.
2004	J2SE with JDK 5.0 was released.

1.2 JAVA Features: [winter 2014,summer 2014,winter 2013,May/June 2012]

1. **Compiled and Interpreted**
   Basically a computer programming language is either compiled or interpreted. Java has both these approach thus making Java a two-stage system. Java compiler (javac) translates Java code (source code/ source file) to Bytecode and Java Interpreter generate machine code that can be directly executed by machine.

2. **Platform Independent and portable**
   Java language provided the portability features. Java programs can be easily run or moved from one computer system to another computer. Changes and upgrades in OS and resources will not force any alteration in Java programs. Java provided the portability in two ways. First way is that, Java compiler (javac) generates the bytecode and that bytecode can be executed on any machine. Second way is, size of primitive data types are machine independent.

3. **Object- oriented**
   Java is pure (truly) object-oriented language. Almost everything is an Object in java programming language. All program code and data are existing in objects and classes.

4. **Robust and secure**

Java is a most strong and more powerful language which provides many securities to make more reliable code as compare to other programming languages. It is design as garbage collected language, which helps the programmers (user) virtually from all memory management problems. Java also support concept of exception handling, which detain serious errors and reduces all kind of threat of crashing the system.

5. **Distributed**
   Sometimes, Java language is also called Distributed language because we built applications on networks which can donate both data and programs. Java applications can open and access from anywhere easily. That means multiple programmers can access multiple remote locations to work together on single task.

6. **Familiar, simple and small**
   Java is very familiar as well as small and simple language. Java does not use pointer and header files, goto statements, etc. It also eliminates operator overloading and multiple inheritance.

7. **Multithreaded and Interactive**
   Java also support multithreading concept. Multithreaded means manage multiple tasks simultaneously. That means we need not wait for the application to complete one task before starting next task. In other word we can say that more than one task are executed in parallel. This feature is helpful for graphic applications.

8. **High performance**
   Performance of java language is actual unexpected for an interpreted language, due to the use of intermediate bytecode. Java architecture is also designed to shrink overheads during runtime. The multithreading improves the execution speed of program.

9. **Dynamic and Extensible**
   Java is also dynamic language. Java is capable of dynamically linking in new class, libraries, methods and objects. Java program is support functions written in other language such as C and C++, known as native methods.

1.3. Concept of Object Oriented Programming Language. [summer 2013]

1. **Object**
   Objects are runtime entities in object oriented method or programming languages. Object is logical as well as physical entity. They may characterize alocation, a bank account, and a table of data or any entry that the program must handle. Each object holds data and code to operate the data.
   For example:

CAR
Data Color Cost
Method Lockit()

Unlockit()
Driveit()
Parkit()

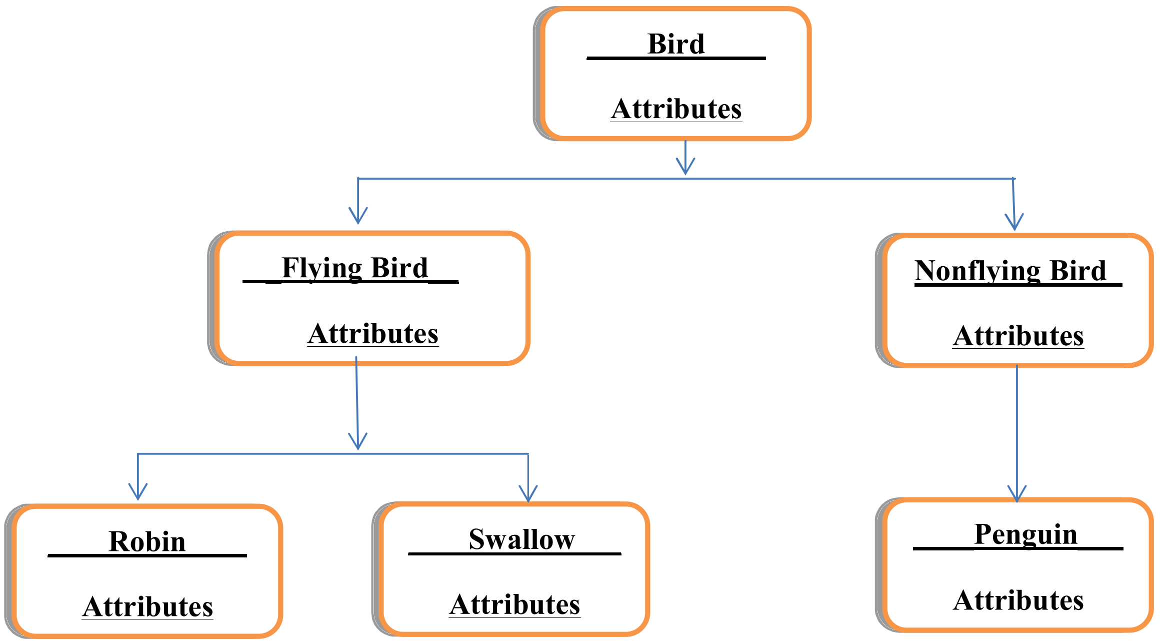
Figure. Representation of object —CAR

2. **Class**
   A class is a logical entity. A class is a set of objects with similar properties (attributes), common behaviour (operations),and common link to other objects. The complete set of data and code of an object can be made a user defined data type with the help of class.

3. **Data abstraction**
   Data abstraction is defined like that the act of representing important description without containing the background details or explanations. When Classes use the concept of abstraction then they are defined as a list of abstract attributes such as size, cost and functions operate on these attributes. Classes use the concepts of data abstraction and it is called as Abstract Data Type (ADT).

4. **Data encapsulation**
   Data Encapsulation means wrapping of data and functions into a single unit. It is most useful feature of class. The data is not easy to get to the outside world and only those functions which are enclosed in the class can access it. These functions provide the boundary between Object‘s data and program. This insulation of data from direct access by the program is called as **Data hiding**.

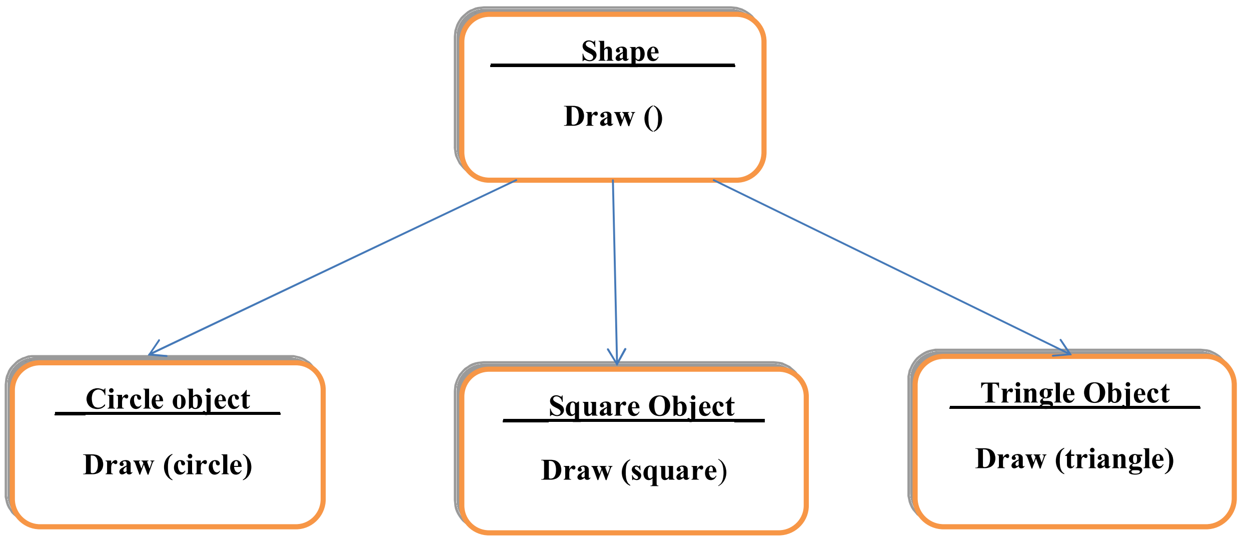
5. **Inheritance**
   Inheritance is the process by which objects of one class can use or get the properties of objects of another class. Inheritance means one class of objects inherits the data and behaviours from another class. Inheritance maintains the hierarchical classification.
   Inheritance provides the important feature of OOP that is reusability. That means we can use some properties of object of that existing class without modification. This is possible deriving anew class from existing one.



Inheritance is a relationship between classes where one class is the parent class of another (derived) class. The derived class holds the properties and behaviour of base class in addition to the properties and behaviour of derived class.

6. Polymorphism

(Poly means -many and morph means -form). Polymorphism means the ability to take more than one form. This means that a general class of operations may be accessed in the same manner even though specific activities associated with each operation may differ. Polymorphism is broadly used in implementing inheritance.



7. Dynamic binding

Binding is defined like that, the linking of a procedure call to the code to be executed in response to the call. Dynamic binding means that the code related with a given procedure call is unknown until the time of the call at run time. Dynamic binding is associated polymorphism and inheritance.

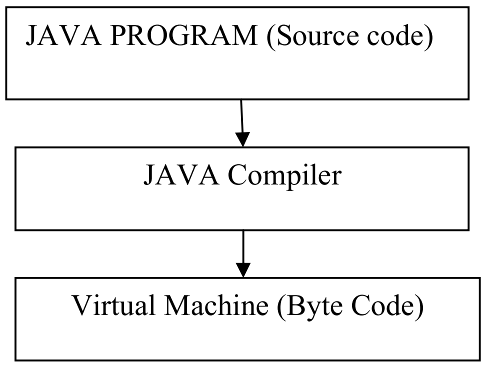
1.4. Comparison between Java and C++

JAVA	C++
Java is a pure object oriented programming language.	C++ is basically C with object oriented extension.
Java does not support multiple inheritances.	C++ supports multiple inheritances.
Java does not support template classes.	C++ has template classes.
Java does not support operator overloading.	C++ supports operator overloading.
Java program compiled into byte code for java virtual machine(JVM)	C++ program can be written to be platform independent.
Java does not use pointer	C++ supports pointer.
Java does not support global variables.	C++ supports Global variables.
No header file in java	Use header file in C++

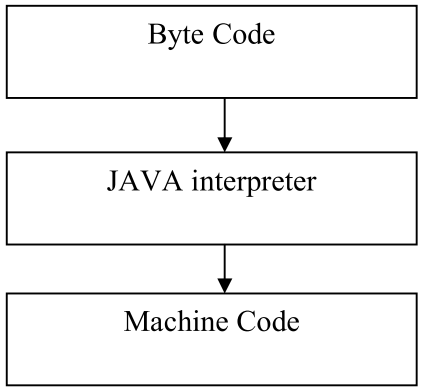
1.5 Java Virtual Machine and Byte Code: [winter 2014, summer 2013]

In all programming language compilers convert the source code (source program) to machine Code (machine program). Same job done by Java Compiler to run a Java program, but the difference is that Java compiler convert the source code (source program) into Intermediate code is called as byte code. This machine is called the Java Virtual machine and it exits only inside the computer memory.

See below figure for process of compilation.



Virtual Machine code is not a machine code. The machine code is generated by java interpreter and this java interpreter is acting as an intermediator between the virtual machine and real machine.



1.6 JDK

The Development tools are part of the system known as Java Development Kit (JDK) and the classes and methods are part of the Java Standard Library (JSL), also known as the Application Programming Interface (API).

Java Development kit (JDK) – The JDK comes with a set of tools that are used for developingand running Java program. It includes:

1.Appletviewer (It is used for viewing the applet)
2.Javac(It is a Java Compiler)

3.Java(It is a java interpreter)
4.Javap(Java disassembler,which convert byte code into program description)
5.Javah(It is for java C header files)
6.Javadoc(It is for creating HTML document)
7.Jdb(It is Java debugger)

For compiling and running the program we have to use following commands:

a) javac (Java compiler)
In java, we can use any text editor for writing program and then save that program with .java extension. Java compiler converts the source code or program in bytecode and interpreter convert .java file in .class file.

Syntax:
```
C:\javac filename.java
```

If my filename is Myfirstprogram.java then the syntax will be

```
C:\javac Myfirstprogram.java
```

b)java(Java Interpreter)
As we learn that, we can use any text editor for writing program and then save that program with .java extension. Java compiler converts the source code or program in bytecode and interpreter convert .java file into .class file.

Syntax:
```
C:\java filename
```

If my filename is Myfirstprogram.java then the syntax will be
```
C:\java Myfirstprogram
```

1.7 Data Type in Java:

There are two types of data types
1. primitive data type
2. non-primitive data type

In primitive data types, there are two categories
1. Numeric primitive data type. e.g. Integer, Floating points
2. Non-numeric primitive datatype e.g. character and Boolean

In non-primitive types, there are three categories
classes
arrays
interface

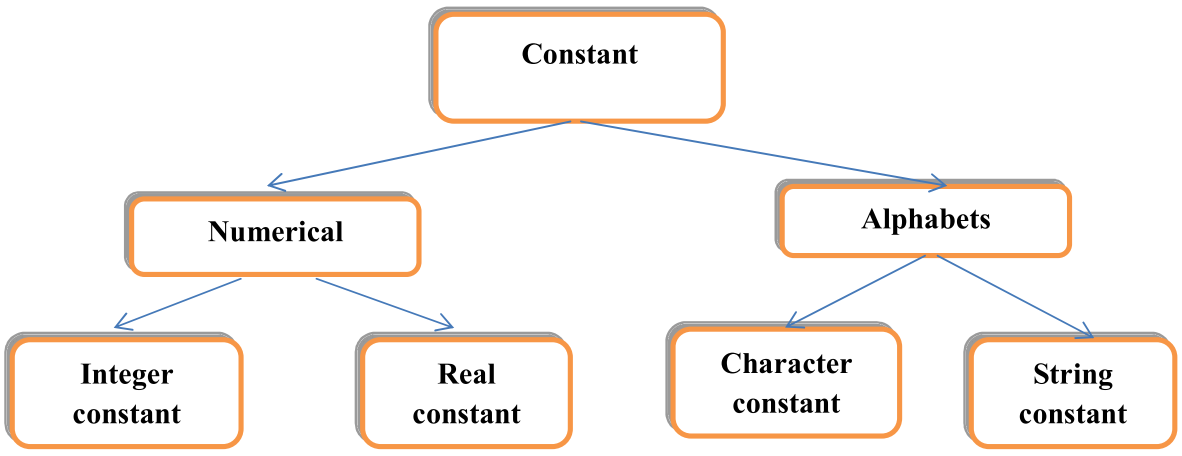
See below table shows data types and their range and size.

Data type	Size in byte	Range
byte	1	-128 to +127
boolean	1	True or False
char	2	A-Z ,a-z, (0 to 9) digits, etc.
short	2	-32768 to +32767
int	4	-2147483648 to 2147483647
long	8	-9223372036854775808 to 9223372036854775807
float	4	-3.4 e38 to +3.4e18
double	8	-1.7e308 to +1.7e308

1.8 Variable name in java OR Naming in java:

- In variable name, use only character (A to Z, a to z) ,digit (0 to 9) and underscores '_'
- Variable name cannot include with space character.
- Variable name does no begin with digit.
- Variable name always start with characters.
- Upper case and lower case count as different variables.
- Keywords not allows as a variable name.

1.9 Constant in java:



(a) Interger Constant:

An Integer constant is denoted with digits.

There are three types of integer as follows:

i) Decimal integer

For example:

11
700
1733

ii) Octal integer

Octal number is denoted from 0 to 7 digits with leading 0.

For example:

010

015
00
0225

ii) Hexadecimal integer

Hexadecimal number is denoted from 0 to 9 digits as well as A to F allow in hexadecimal number. And it is preceded by 0X or 0x.
For example:

0x88
0x8A
0x15A

(b) Real Constant:

Real Constant means any number with decimal point.
For example:

3.14
0.55
- 11.23

(c) Character constant:

Character constant means single character with denoted by pair of single coute.
For example:

‘A‘
‘7‘
‘0’

(d) String Constant:

String constant means group of characters within pair of double coute.
For example:

“Welcome to Java Programming”
“Hello world”

1.10 Escape Sequences OR Backslash character constant:

Symbol of Character constant	Meaning
‘\n’	New line
‘\b’	Back space
‘\t’	Tab (Group of five space)
‘\\’	Back slash
‘\’’	Single contention mark
‘\”’	Double contention mark

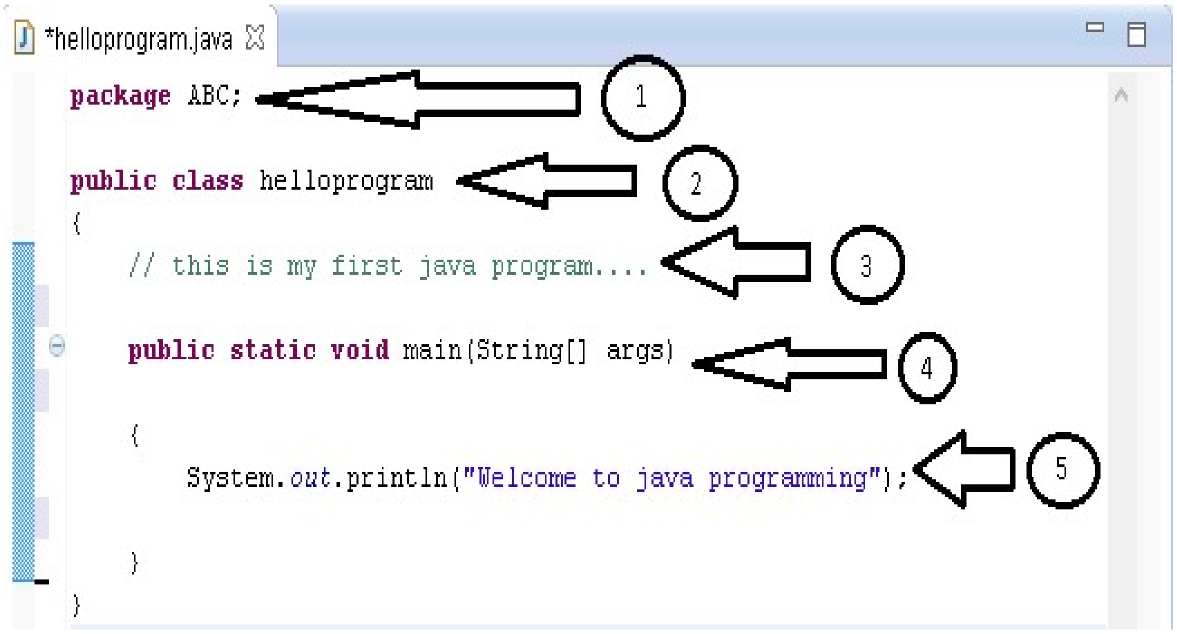
1.11 Keyword in java

Keywords are important part of java. Java language has number of keywords reserved. Let’s we see it below table.

abstract	long	import	transient
char	throw	protected	native
catch	private	class	null
boolean	package	throws	const

default	static	byte	new
finally	break	else	case
do	double	float	extends
implements	this	Final	int
if	volatile	public	instanceof
return	interface	short	assert
try	void	continue	const
for	while	goto	
Switch	synchronized	super	

1.12 Java Program Structure:



```
package ABC;

public class helloprogram
{
    // this is my first java program....

    public static void main(String[] args)

    {
        System.out.println("Welcome to java programming");

    }
}
```

1. **package ABC;**
   It is package declaration statement. The package statement defines a name space in which classes are stored. Package is used to organize the classes based on functionality.

2. **public class helloprogram**
   This is access modifier keyword which tells compiler access to class. Various values of access modifiers can be public, protected, and private or default (no value).

3. **Comments**
   // double forward slash use for single line comments
   /* -----*/ use for multiple line comment.
   Comment lines not execute by java compiler.

4. **public static void main(String [] args)**

public: Access Modifier
static: static is reserved keyword which means that a method is accessible and usable even though no objects of the class exist.

void: This keyword declares nothing would be returned from method. Method can return any primitive or object.
Method content inside curly braces. { }.

5. **System.out.println("welcome to java programming");**
   System: It is name of Java utility class.
   out: It is an object which belongs to System class.
   println: It is utility method name which is used to send any String to console.

1.13 Operators in java

List of Operators see below:

- Arithmetic operators:
- Logical operators:
- Relational operators:
- Assignment operators:
- Bitwise operators:
- Conditional operators:
- Increment and decrement operators:

(A) Arithmetic Operator:

operator	description
+	addition
-	subtraction
*	multiplication
/	division
%	modulo

For example:

```
x = 16 % 3;
```

Result in variable x containing the value 1, since dividing 16 by 3 results in 5, with a remainder of 1.

Example 1:

```
packageABC;
publicclass ArithmeticOp
{
        publicstaticvoid main(String[] args)
        {
                int a,b,add,sub,mul,div,mod,avg;
                a=16;
                b=3;
                add=a+b;
                sub=a-b;
                mul=a*b;
                div=a/b;
                mod=a%b;
                avg=(a+b)/2;
                System.out.println("addition      =" +add);
                System.out.println("subtraction   =" +sub);
                System.out.println("multiplication =" +mul);
                System.out.println("division      =" +div);
                System.out.println("modules       =" +mod);
                System.out.println("average       =" +avg);
        }
}
```

Output:

```
addition      =19
subtraction   =13
multiplication =48
division      =5
modules       =1
average       =9
```

.

(B) Logical Operator Or Short Circuit Operator: [summer 2014,summer 2013,june 2011]

Logic operators **&& (and)** and || **(or)** are used when evaluating two expressions to obtain a single result. They correspond with Boolean logic operations AND and OR respectively. The result of them depends on the relation between its two operands:

Example: And Operator

if(condition 1 && condition 2)
{
Statements;
}
……
Statement x;
	If both conditions are become true then after control goes inside body and execute statements.
	If even any one condition becomes false then skip statement and then direct execute statement x.

See below table, Logical operator work like Boolean algebra And operator.

A	**B**	**c**	Note: here a and b are consider as a condition and c is consider as an output action. Here 0 means condition becomes **false** and 1 means conditions becomes **true.** And you can see result (c) in side table.
0	0	**0**	
0	1	**0**	
1	0	**0**	
1	1	**1**	

Example 1:

```
//find maximum number from given three number using AND (&&) operator
packageABC;
publicclassDemoLogical
{
        publicstaticvoid main(String[] args)
        {
                inta, b, c;
```

```
                    a=16;
                    b=3;
                    c=20;
                    if(a>b && a>c)
                    {
                        System.out.println("a is max" + a);
                    }
                    if(b>a && b>c)
                    {
                        System.out.println("b is max" + b);
                    }
                    if(c>a && c>b)
                    {
                        System.out.println("c is max" + c);
                    }
          }
}
```

```
Output:
c is max=20
```

Example: **OR Operator (||)**

if(condition 1 || condition 2)
{
Statements;
}
……
Statement x;
	Here, either Condition 1 becomes true or condition 2 becomes true, then statements will execute then after statements x will execute.
	If both are conditions are become false then skip statements which are inside the body of if statements and then execute statements X.

See below table, OR operator work like Boolean algebra OR operator.

A	**B**	**c**	Note: here a and b are consider as a condition and c is consider as an output action. Here 0 means condition becomes **false** and 1 means conditions becomes **true.** And you can see result (c) in side table.
0	0	**0**	
0	1	**1**	
1	0	**1**	
1	1	**1**	

(C) Relational Operators:

Two expressions can be compared using relational and equality operators. For example, to know if two values are equal or if one is greater than the other.

operator	**description**
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Example 1:

```
//demo of relational operator
packageABC;

publicclassRelationalOp
{
        publicstaticvoidmain(String[] args)
        {
                inta, b;
                a=16;
                b=3;
                System.out.println("a>b   " + (a>b));
                System.out.println("a>=b  " + (a>=b));
                System.out.println("a<b   " + (a<b));
                System.out.println("a<=b  " + (a<=b));
                System.out.println("a!=b  " + (a!=b));
                System.out.println("a==b  " + (a==b));
        }
}
```

Output:

```
a>b   true
a>=b  true
a<b   false
a<=b  false
a!=b  true
a==b  false
```

(D) Assignments Operator:

The assignment operator assigns a value into a variable.

```
x = 5;
```

This statement assigns the integer value 5 to the variable x.

Example 1:

```
//demo of assignment operator
package ABC;
publicclasshelloprogram
{
        publicstaticvoid main(String[] args)
        {
                int a, b;                // a=?,  b=?
                  a = 10;                  // a=10, b=?
                  b = 4;                  // a=10, b=4
                  a = b;                  // a=4,  b=4
                  b = 7;                  // a=4,  b=7
 System.out.println("Value of a=" +a);   //print value of a
  System.out.println("Value of b=" +b);   // print value of b
}
}
```

Output:

```
Value of a=4
Value of b=7
```

(E) Bitwise Operator: [summer 2013]

Bitwise operators modify variables considering the bit patterns that represent the values they store.

operator	description
&	Bitwise AND
|	Bitwise inclusive OR
^	Bitwise exclusive OR(XOR)
~	Unary complement (bit inversion)
<<	Shift bits left
>>	Shift bits right

Example 1: bitwise AND

```
package ABC;

publicclass DemobitwiseAnd
{
  publicstaticvoid main(String[] args)
        {
                int a,b,c;
                a=10;
                b=7;
                c=a & b;
                System.out.println("ans is =" +c);

        }
}
```

Description:
a=10 and b=7 first convert into binary value.

```
00001010     this is 10 value binary
00000111     this is 5 value binary
```

Now And (&&) operator apply to those above binary value. And we will get value of c variable.

```
          00001010
          00000111
AND       00000010    this value is
convert  into decimal
answer is C=2
```

Output:

```
ans is =2
```

Example 2: bitwise OR

```
package ABC;

publicclass Demobitwiseor
{
publicstaticvoid main(String[] args)
        {
        int a,b,c;
        a=10;
        b=7;
        c=a | b;
        System.out.println("ans is =" +c);
        }
}
```

Description:
a=10 and b=7 first convert into binary value.

```
00001010     this is 10 value binary
00000111     this is 5 value binary
```

Now OR operator apply to those above binary value. And we will get value of c variable.

```
        00001010
        00000111
OR      00001111    this value is convert
into decimal
```

answer is C=15

Output:
ans is =15

Example 3: bitwise XOR

```
package ABC;
publicclass DemobitwiseXor
{
publicstaticvoid main(String[] args)
{
        int a,b,c;
        a=10;
        b=7;
        c=a ^ b;
        System.out.println("ans is =" +c);
}
}
```

Here we have declared a, b, c variable. Then assign value 10 into var a, and assign value 7 into variable b.
Now variables a and b convert into binary. See below.

A	00001010
B	00000111
XOR (c)	00001101

So after perform XOR operation, the result is assign in to variable C, so here c variables value becomes 14.
C=13.

Output:
ans is =13

Example 4: bitwise shift left

```
package ABC;

publicclass demobitwiseshiftleft
{
publicstaticvoid main(String[] args)
        {
        int a,b;
        a=20;
        b=a<<1;
        System.out.println("ans is =" +b);      }
}
```

a =20; we have first convert into binary value.
00010100.
Now each bit shift one bit left side.
00101000.
After we will get bit position like this.
Now convert above bit patent into decimal and get result b=40;

Output:
ans is = 40

Example 5: bitwise shift right

```
package ABC;

publicclass BitwiseShiftright
{
publicstaticvoid main(String[] args)
        {
        int a,b;
        a=20;
        b=a>>1;
        System.out.println("ans is =" +b);
        }
}
```

a =20; we have first convert into binary value.
00010100.
Now each bit shift one bit right side.
After we will get bit position like this.
00001010
Now convert above bit patent into decimal and get result b=10;

Output:

```
ans is =10
```

(F) Conditional Operator Or Ternary Operator:

The conditional operator evaluates an expression, returning one value if that expression evaluates to true, and a different one if the expression evaluates as false

Syntax:

```
condition? result1: result2;
```

Example 1:

```
//demo of conditional operator
package ABC;
publicclass ConditinalOP
{
        publicstaticvoidmain(String[] args)
        {
                int a,b,c;
                a=10;
                b=7;
                c = (a>b)? a: b;
                System.out.println("maximum number is=" +c);
        }
}
```

Output:

```
maximum number is=10
```

(G) Increment or Decrement Operator:

The increase operator (++) and the decrease operator (--) increase or reduce by one the value stored in a variable. They are equivalent to +=1 and to -=1, respectively.

Example 1	Example 2

x = 3;
y = ++x;
// x contains 4, y contains 4
	x = 3;
	y = x++;
	// x contains 4, y contains 3

Example 1:

```
// demo of prefix-increment and postfix- increment operator
package ABC;
publicclass helloprogram
{
        publicstaticvoid main(String[] args)
        {
                int a,b,c;
                a=10;                                   //a=10, b=?, c=?
                a++;                                  //a=11, b=?,c=?
                System.out.println("value of a="+a);
                b=a++;                                 //a=12, b=11,c=?
                c=++a;                                 //a=13, b=11,c=13?
                System.out.println("value of a="+a);   //a=13 print
                System.out.println("value of b="+b);   //b=11 print
```

```
                    System.out.println("value of c="+c);           //c= 13 print
}
}
```

```
Output:
value of a=11
value of a=13
value of b=11
value of c=13
```

Example 2:

```
// demo of prefix-decrement and postfix- decrement operator
package ABC;
publicclass helloprogram
{
        publicstaticvoid main(String[] args)
        {
                int a,b,c;
                a=10;                                          //a=10, b=? , c=?
                a--;                                         //a=9, b=? , c=?
                System.out.println("value of a="+a);
                b=a--;                                          //a=8, b=9, c=?
                c=--a;                                          //a=7, b=9, c=7?
                System.out.println("value of a="+a);           //a=7 print
                System.out.println("value of b="+b);           //b=9 print
                System.out.println("value of c="+c);           //c=7 print
 }
}
```

```
Output:
value of a=9
value of a=7
value of b=9
value of c=7
```

1.14 Selection statements / Control structure:

1. if statements
2. if else statements
3. nested if else statements
4. if else ladder statements
5. switch statements

1. if statements:

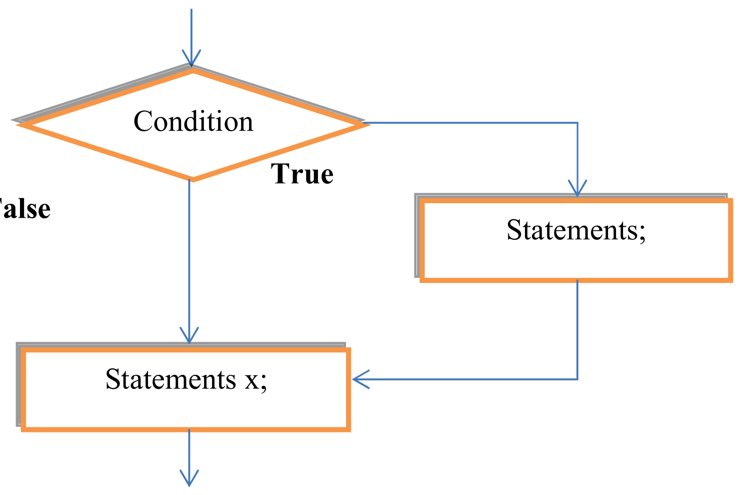
java programing has support conditional statements; here we need some time skip number of statements depending on programing logic.so we needed used conditional statements.
Now de discuses if statements see below. If expression or condition becomes true then control goes into body of if statement and then executed statement x; If suppose

expression or condition becomes false then control dose not goes inside body of if statements, and then after it execute statement x;

Syntax:

```
if (expression)
{
        Block of statements;
}
…..
Statement x;
```

Flowchart:



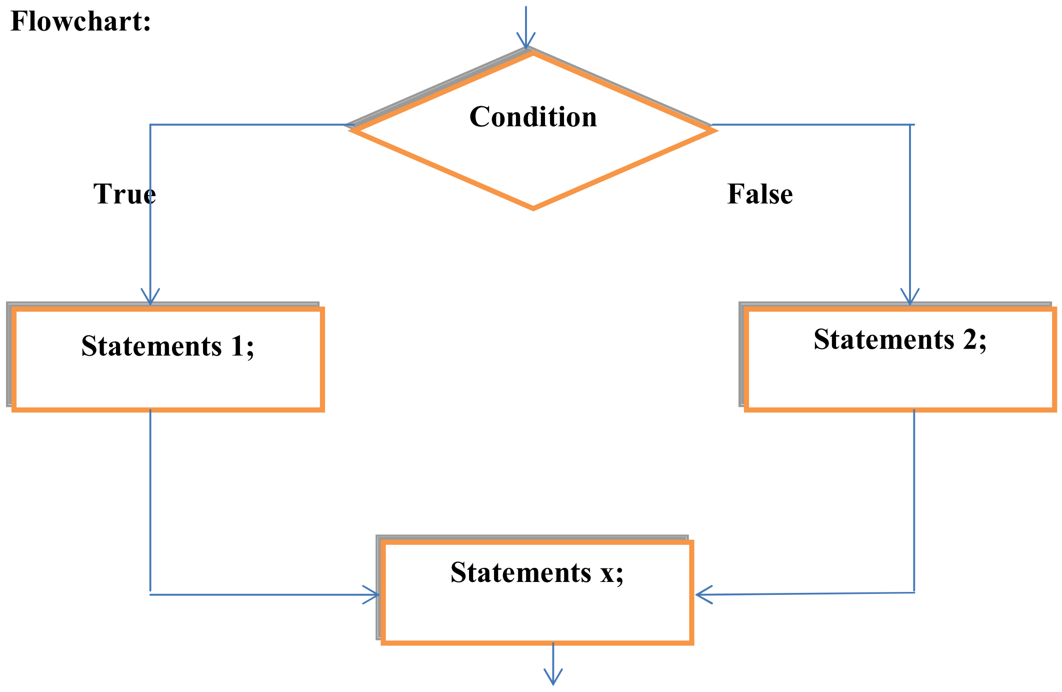
2. if else statements:

Now de discuses if-else statements see below. If expression or condition becomes true then control goes into if body and then executed statement 1; If suppose expression or condition becomes false then control goes inside body of else part and executed statements 2, and then after it execute statement x;

Syntax:

```
if(condition)
{
        statement 1;
}
else
{
        statement 2;
}
Statement x;
```

Flowchart:



Example 1:

```
// demo of maximum number form given two number using if else statements.
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass maximum
{
publicstaticvoid main(String[] args) throws NumberFormatException, IOException
        {
                int a ,b;

        DataInputStream s1=new DataInputStream(System.in);

        System.out.println("Enter value of A and B");

        a=Integer.parseInt(s1.readLine());  //cast in interger of enter number a
        b=Integer.parseInt(s1.readLine()); //cast in interger of enter number b

if(a>b)
                {
                        System.out.println("A is max=" +a );
                }
        else
                {
```

```
                              System.out.println("B is max=" +b );
                    }

          }
}
```

Output:

```
Enter value of A and B
12
15
B is max=15
```

Example 2:

```
//demo of find out enter number is even or odd.
packageABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass Evenodd
{
publicstaticvoid main(String[] args) throws NumberFormatException, IOException
          {
                    int a ;
                    DataInputStream s1=new DataInputStream(System.in);
                    System.out.println("Enter value of A");
                    a=Integer.parseInt(s1.readLine());

                    if(a % 2 == 0)
                    {
                              System.out.println("given number is even=" + a);
                    }
                    else
                    {
                              System.out.println("given number is odd=" + a);
                    }
          }
}
```

Output:

```
Enter value of A
12
given number is even=12
```

3. **nested if else statements:**

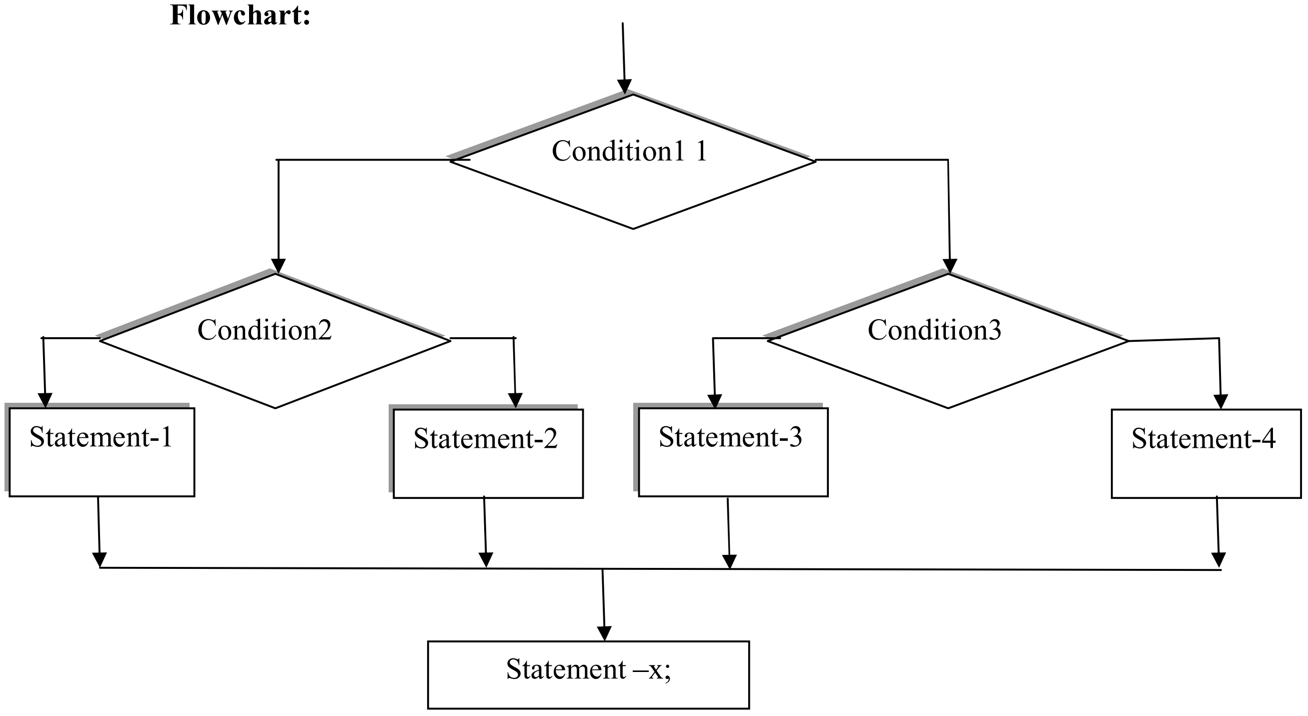
   Nested if statements means one if statements or more than one if statements are inside other if condition.

   Syntax:

```
if(condition 1)
{
```

```
            if(condition 2)
            {
                  if(condition 3)
                  {
                     Statements;
                  }
            }
}
else
{
             if(condition)
             {
                   if(condition)
                   {
                       Statements;
                   }
             }
}
```

Flowchart:



Example:

```
//Demo of nested if statements.
package ABC;
import java.io.DataInputStream;
import java.io.IOException;

publicclass Maximumnumber
{
public staticvoid main(String[] args) throws NumberFormatException, IOException
```

```
        {
                inta,b,c ;

             DataInputStream s1=new DataInputStream(System.in);
            System.out.println("Enter value of A,B and C variable");

        a=Integer.parseInt(s1.readLine());
        b=Integer.parseInt(s1.readLine());
        c=Integer.parseInt(s1.readLine());

if(a > b)
                {
                        if(a>c)
                        {
                                System.out.println("A is max=" + a);
                        }
                        else
                        {
                                System.out.println("C is max=" + c);
                        }
                }
                else
                {
                        if(b>c)
                        {
                                System.out.println("B is max=" + b);
                        }
                        else
                        {
                                System.out.println("C is max=" + c);
                        }
                }

        }
}
```

Output:

```
Enter value of A,B and C variable
10
20
30
C is max=30
```

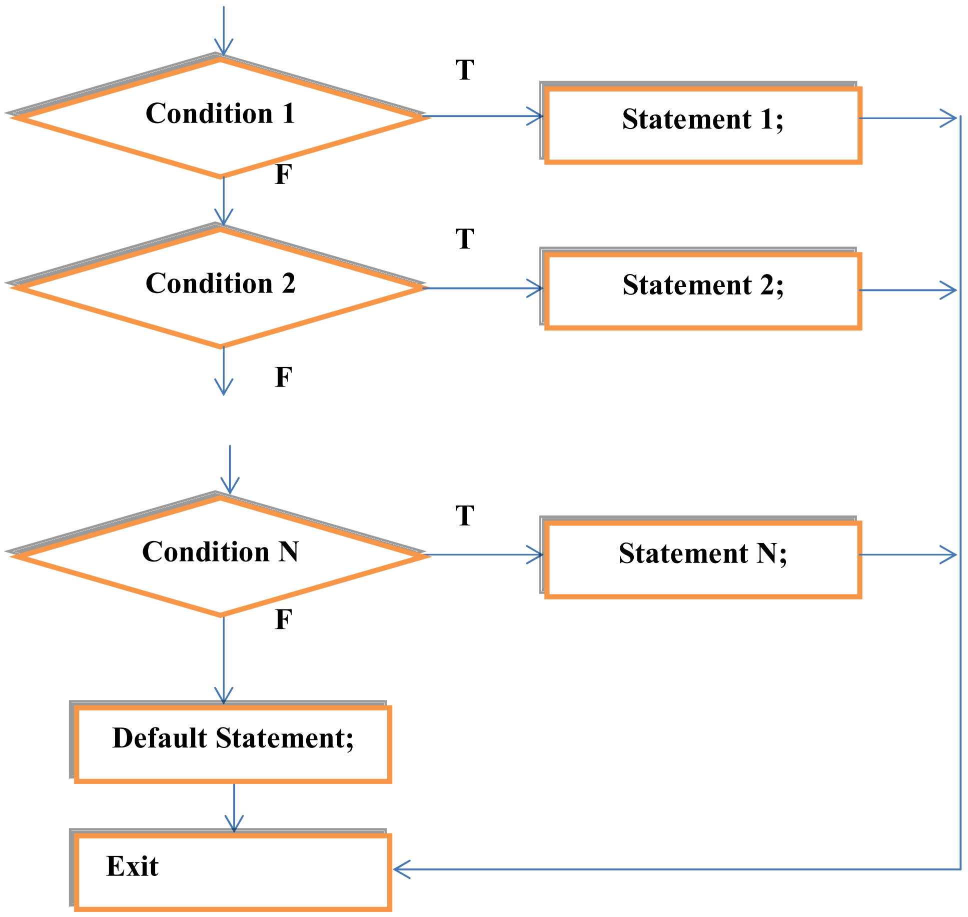
4. **if else ladder statements:**

Now we discuss if-else ladder statements see below. If first condition becomes true then control executed statement 1 and exit from if else ladders statements. If first condition becomes false then control does to check condition 2. If all condition are become false then control goes to else part and execute default statement and exit from the if else ladder statements.

Syntax:

```
if(condition 1)
{
            Statement1 ;
}
else if(condition 2)
{
            Statement2 ;
}
……
……
else if (condition n)
{
              Statement n ;
}
else
{
               Default Statement ;
}
```

Flowchart:



Example:

```
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass DemoElseifLadder
{
publicstaticvoid main(String[] args) throws NumberFormatException, IOException
        {
                int a ;
                DataInputStream s1=new DataInputStream(System.in);
                System.out.println("Enter any number");
                a=Integer.parseInt(s1.readLine());

                if(a==1)
                {
                        System.out.println("Sunday");
                }
                elseif (a==2)
                {
                        System.out.println("Monday");
                }
                elseif(a==3)
                {
                        System.out.println("Tuesday");
                }
                elseif (a==4)
                {
                        System.out.println("Wednesday");
                }
                elseif(a==5)
                {
                        System.out.println("Thursday");
                }
                elseif(a==6)
                {
                        System.out.println("Friday");
                }
                elseif(a==7)
                {
                        System.out.println("Saturday");
                }
                else
                {
                        System.out.println("Please enter number between 1 to 7");
```

```
                    }
            }
}
```

Output:

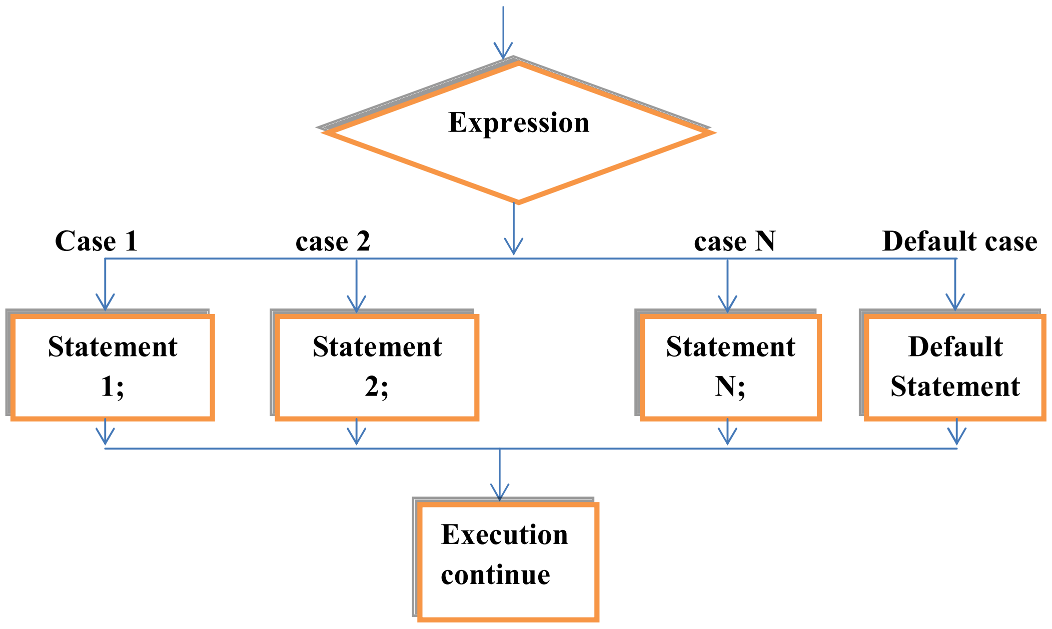
```
Enter any number
5
Thursday
```

5. switch statements:

Syntax:

```
switch( expression or variable or value)
{
            case 1: statement1 ;
                         break;
             case 2: statement2 ;
                        break;
                 ……….
                 ……......
            case n: statement n ;
                         break;
           default:  default statements ;
                          break;
}
```

Flowchart:



Example:

```
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass Demoswitch
{
publicstaticvoid main(String[] args) throws NumberFormatException, IOException
	{
		int a ;
		DataInputStream s1=new DataInputStream(System.in);
		System.out.println("Enter any number");
		a=Integer.parseInt(s1.readLine());

		switch(a)
		{
		case1:  System.out.println("Sunday");
		break;
		case2:  System.out.println("Monday");
break;
		case3:  System.out.println("Tuesday");
		break;
		case4:  System.out.println("Wednesday");
break;
		case5:  System.out.println("Thursday");
break;
		case6:  System.out.println("Friday");
break;
		case7:  System.out.println("Saturday");
break;
		default: System.out.println("Please enter number between 1 to 7");
		}
	}
}
```

Output:

```
Enter any number
12
Please enter number between 1 to 7
```

1.16 Iteration Statements (Looping):

There may be a situation, when you need to execute a block of code several numbers of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Java provided various control structures that allow for more complicated execution paths.

1. while loop
2. do while loop
3. for loop
4. Enhanced for loop

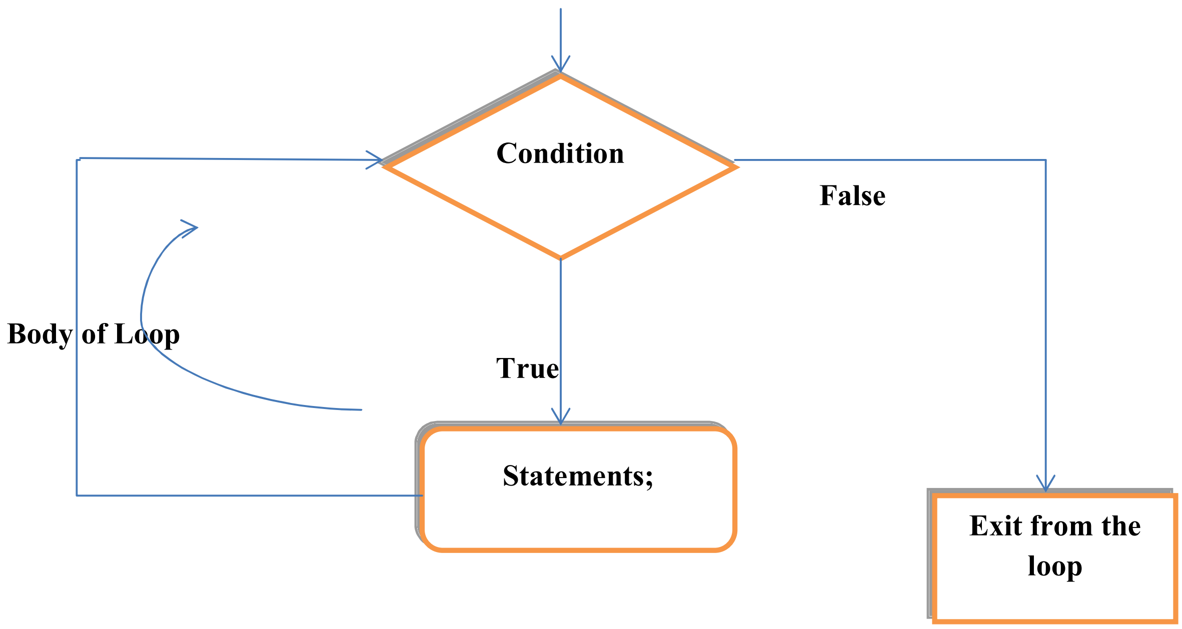
1. while loop

The while loops checks whether the condition is true or not. If it is true, code/s inside the body of while loop is executed, that is, code/s inside the braces { } are executed. Then again the condition is checked whether condition is true or not. This process continues until the condition becomes false.

Syntax:

```
while ( condition)
{
            Statements;
            Increment or decrements;
}
```

Flowchart:



Example 1:

```
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass Demowhile
{
publicstaticvoid main(String[] args)throws NumberFormatException, IOException
```

```
        {
                int n ,i=1;
                DataInputStream s1=new DataInputStream(System.in);
                System.out.println("Enter any number");
                n=Integer.parseInt(s1.readLine());

                while(i<=n)
                {
                        System.out.println("number is=" +i);
                        i++;
                }
        }
}
```

Output:

```
Enter any number
5
number is=1
number is=2
number is=3
number is=4
number is=5
```

Example 2:

```
//sum of n interger number using while loop
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass SumofNnumber
{
        publicstaticvoid main(String[] args) throws NumberFormatException, IOException
        {
                int n ,i=1,sum=0;
                DataInputStream s1=new DataInputStream(System.in);
                System.out.println("Enter any number");
                n=Integer.parseInt(s1.readLine());

                while(i<=n)
                {
                        sum = sum + i;
                        i++;
                }
                System.out.println("Sum is=" +sum);
        }
}
```

Output:

```
Enter any number
5
Sum is=15
```

Example 3:

```
//find out factorial of give number
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass DemoFactorial
{
publicstaticvoid main(String[] args)throws NumberFormatException, IOException
	{
		intn ,i=1,fact=1;
		DataInputStream s1=new DataInputStream(System.in);
		System.out.println("Enter any number");
		n=Integer.parseInt(s1.readLine());

		while(i<=n)
		{
			fact = fact * i;
			i++;
		}
		System.out.println("Factorial is=" + fact);
	}
}
```

Output:

```
Enter any number
5
Factorial is=120
```

Example 4:

```
// Fibonacci series
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass Fibo
{
publicstaticvoid main(String[] args)throws NumberFormatException, IOException
	{
		int n ,i=1,a=0,b=1,c;
		DataInputStream s1=newDataInputStream(System.in);
		System.out.println("Enter any number");
		n=Integer.parseInt(s1.readLine());
		System.out.print(" "+a);
		System.out.print(" "+b);
		while(i<=n)
		{
			c =a+b;
```

```
                    System.out.print(" "+c);
                    a=b;
                    b=c;
                    i++;
                }
        }
}
```

Output:

```
Enter any number
8
 0   1 1 2 3 5 8 13 21 34
```

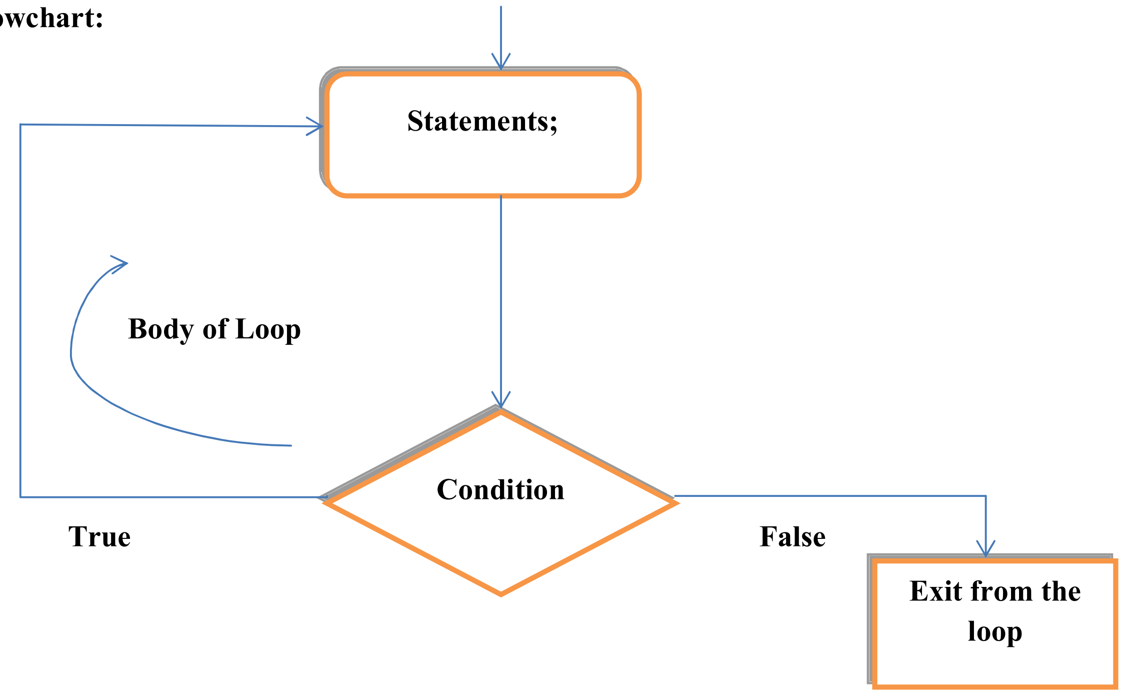
2. do while loop

In do...while loop is very similar to while loop. Only difference between these two loops is that, in while loops, condition is checked at first but, in do...while loop code is executed at first then the condition is checked. So, the code are executed at least once in do...while loops. At first codes inside body of do is executed. Then, condition is checked. If it is true, code/s inside body of do is executed again and the process continues until condition becomes false. Note: there is semicolon in the end of while (); in do...while loop.

Syntax:

```
do
{
        Statements;
}
while(condition);
```

Flowchart:



Example:

```
//print n integer number using do while loop
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass helloprogram
{
publicstaticvoid main(String[] args) throws NumberFormatException, IOException
        {
                int n,i;
                DataInputStream s1=new DataInputStream(System.in);
                System.out.println("Enter any number");
                n=Integer.parseInt(s1.readLine());
        i=0;
do
      {
              System.out.println("Number is=" + i );
              i++;
      }while(i<=n);
 }
}
```

Output:

```
Enter any number
5
Number is=0
Number is=1
Number is=2
Number is=3
Number is=4
Number is=5
```

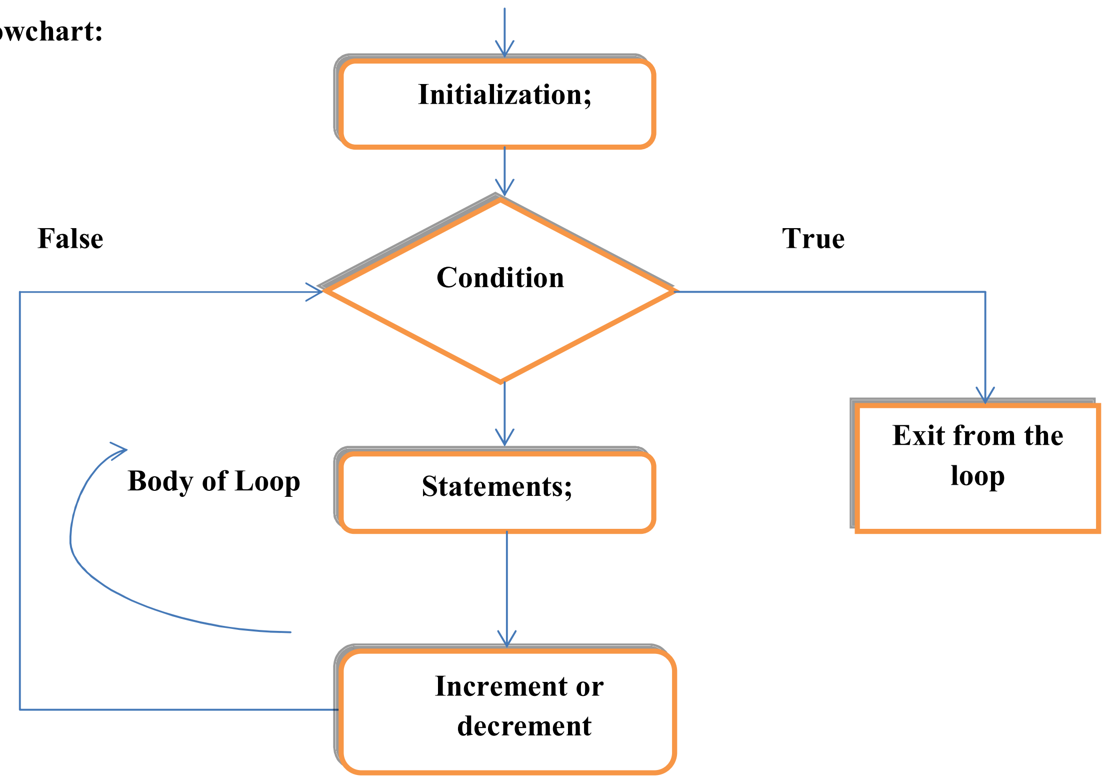
3. for loop

The initialization statement is executed only once at the beginning of the for loop. Then the condition is checked by the program. If the condition is false, for loop is terminated. But if condition is true then the code/s inside body of for loop is executed and then update expression is updated. This process repeats until test expression is false.

Syntax:

```
for (initialization; condition; increment/decrement)
{
……
……
Statements;
……
……
}
```

Flowchart:



Example 1:

```
//print n integer number
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass DemoForLoop
{
publicstaticvoid main(String[] args)throws NumberFormatException, IOException
        {
                int n ,i=1;
                DataInputStream s1=newDataInputStream(System.in);
                System.out.println("Enter any number");
                n=Integer.parseInt(s1.readLine());

                for(i=1;i<=n;i++)
                {
                        System.out.println("number is=" +i);
                }
        }
}
```

Output:

```
Enter any number
```

```
5
number is=1
number is=2
number is=3
number is=4
number is=5
```

Example 2:

```
//sum of n integer number
package ABC;

import java.io.DataInputStream;
importjava.io.IOException;

publicclass DemoOfSumofnNumber
{
Publicstaticvoid main(String[] args)throws NumberFormatException, IOException
        {
                int n ,i=1,sum=0;
                DataInputStream s1=new DataInputStream(System.in);
                System.out.println("Enter any number");
                n=Integer.parseInt(s1.readLine());

                for(i=1;i<=n;i++)
                {
                        sum= sum + i;
                }
                System.out.println("sum of given number is=" +sum);
        }
}
```

Output:

```
Enter any number
5
sum of given number is=15
```

Example 3:

```
//find out factorial of given number
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclass  Demofactorial
{
publicstaticvoid main(String[] args)throwsNumberFormatException, IOException
        {
                int n ,i=1,fact=1;
                DataInputStream s1=new DataInputStream(System.in);
                System.out.println("Enter any number");
                n=Integer.parseInt(s1.readLine());
                for(i=1;i<=n;i++)
                {
                        fact= fact*i;

                }
                System.out.println("Factorial is=" +fact);
        }
}
```

Output:

```
Enter any number
5
Factorial is=120
```

Example 4:

```
// Fibonacci series
package ABC;

import java.io.DataInputStream;
import java.io.IOException;

publicclassfibo
{
publicstaticvoid main(String[] args)throws NumberFormatException, IOException
        {
                int n ,i=1,a=0,b=1,c;
                DataInputStream s1=new DataInputStream(System.in);
                System.out.println("Enter any number");
                n=Integer.parseInt(s1.readLine());
                System.out.print(" "+a);
                System.out.print(" "+b);
                for(i=1;i<8;i++)
                {
                        c =a+b;
                        System.out.print(" "+c);
                        a=b;
                        b=c;
                }
        }
}
```

Output:

```
Enter any number
7
 0 1 1 2 3 5 8 13 21
```

4. **Enhancedforloop / for each loop:**

In java the enhanced for loop was built for mainly used in arrays.

Syntax:

```
for(declaration : expression)
{
 Body of loop
}
```

Example:

```
publicclass EnhancedForLoop
{
    publicstaticvoid main(String[] args) {
int[] a={10,20,30,40,50};
String [] str={ "hi", "How","are", "you", "i" ,"am", "fine"};
        System.out.println("enhanced loop used for integer array ");
for(inti :a)
        {
        System.out.println(i);
        }
System.out.println("enhanced loop used for String array ");
for(String  i :str)
        {
    System.out.println(i);
        }
}
}
```

Output:

```
enhanced loop used for integer array
10
20
30
40
50
enhanced loop used for String array
hi
How
are
you
i
am
fine
```

1.17 Asked Question in GTU Papers:

Q 1. Explain feature of java.   [Winter 2014, summer 2014, winter 2013, May June 2012]
Q 2. Explain short circuit operator and shift operator.[June 2011]
Q 3. List of OOP characteristics.[Summer 2013]
Q 4. Explain short circuit operator and shift operator.[Summer 2013]