Q.1 . What is an Operator ? Mention Java operators.
Ans :
Operator is a symbol which is used to perform operations. For example: +, -, *, / etc.

There are many types of operators in Java which are given below:

- Unary Operator,
- Arithmetic Operator,
- Shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.

Operator Type	Category	Precedence
Unary	postfix	expr++ expr--
	prefix	++expr --expr +expr -expr ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<< >> >>>
Relational	comparison	< > <= >= instanceof
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :

Assignment	assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Java Unary Operator

The Java unary operators require only one operand. Unary operators are used to perform various operations i.e.:

- incrementing/decrementing a value by one
- negating an expression
- inverting the value of a boolean

Java Arithmetic Operators

Java arithmatic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations.

```
class OperatorExample
{
public static void main(String args[])
{
int a=10;
int b=5;
System.out.println(a+b);//15
System.out.println(a-b);//5
System.out.println(a*b);//50
System.out.println(a/b);//2
System.out.println(a%b);//0
}
}
```

Java Left Shift Operator

The Java left shift operator << is used to shift all of the bits in a value to the left side of a specified number of times.

```
class OperatorExample
{
public static void main(String args[])
{
System.out.println(10<<2);//10*2^2=10*4=40
System.out.println(10<<3);//10*2^3=10*8=80
System.out.println(20<<2);//20*2^2=20*4=80
System.out.println(15<<4);//15*2^4=15*16=240
}
}
```

Java Right Shift Operator

The Java right shift operator >> is used to move left operands value to right by the number of bits specified by the right operand.

```
class OperatorExample
{
public static void main(String args[])
{
System.out.println(10>>2);//10/2^2=10/4=2
System.out.println(20>>2);//20/2^2=20/4=5
System.out.println(20>>3);//20/2^3=20/8=2
}
}
```

Java Ternary Operator

Java Ternary operator is used as one liner replacement for if-then-else statement and used a lot in Java programming. it is the only conditional operator which takes three operands.

Java Assignment Operator

Java assignment operator is one of the most common operator. It is used to assign the value on its right to the operand on its left.

Java Assignment Operator

Java assignment operator is one of the most common operator. It is used to assign the value on its right to the operand on its left.

Q. 2. Write short note son :

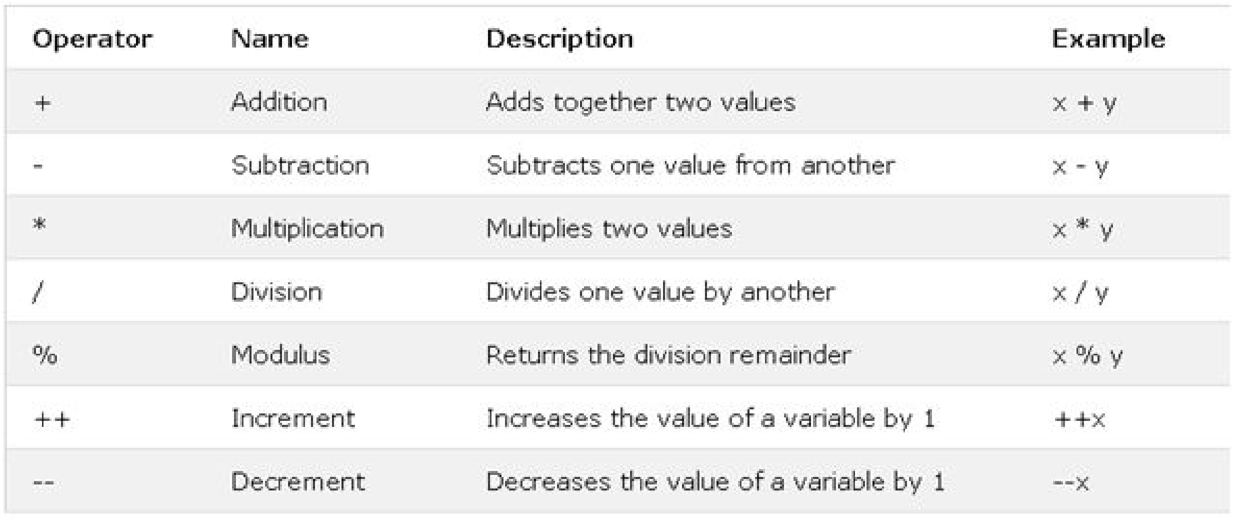
Arithmetic operators, Bitwise operators, Relational operators, Boolean logical

operators, Assignment operator and Conditional operator.

Ans :

Arithmetic operators:

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. A value used on either side of an operator is called an operand.For example, in below statement the expression 47 + 3, the numbers 47 and 3 are operands. The arithmetic operators are examples of binary operators because they require two operands. The operands of the arithmetic operators must be of a numeric type.



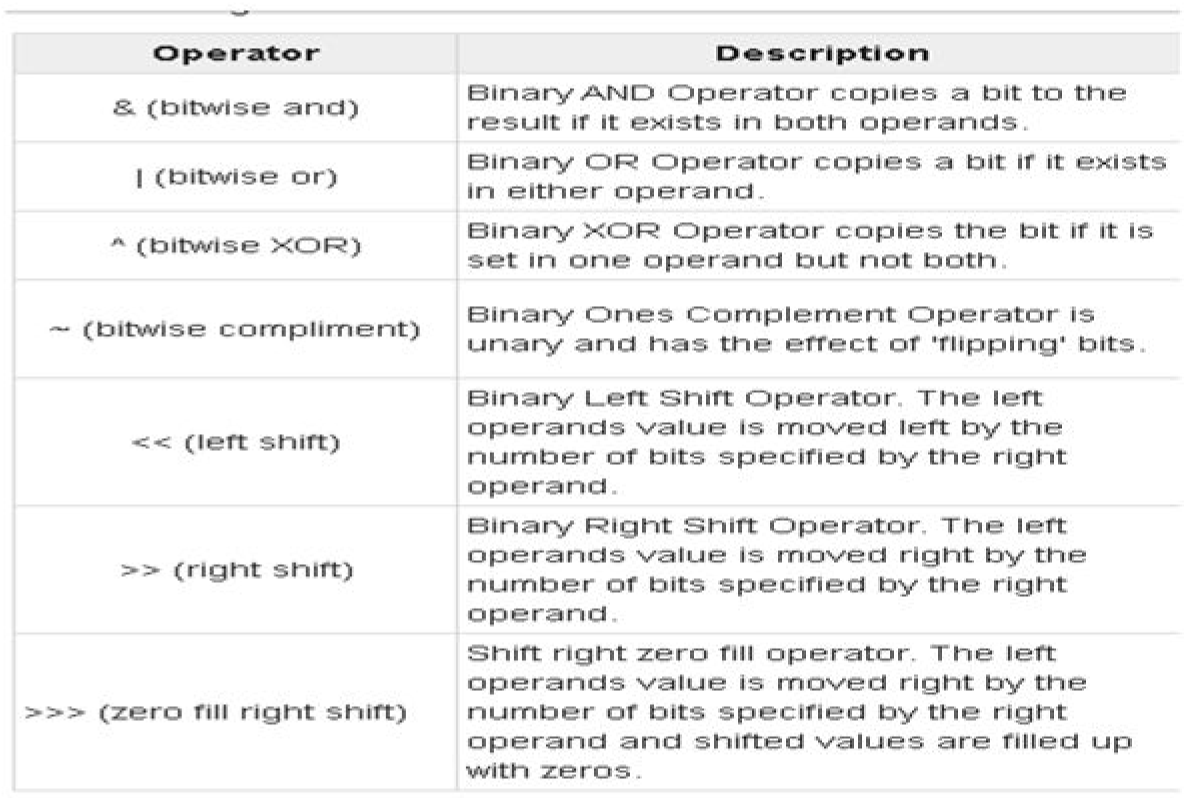
Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Bitwise Operator :

Java defines several bitwise operators, which can be applied to the integer types, long, int, short, char, and byte.

Bitwise operator works on bits and performs the bit-by-bit operation.

The following table lists the bitwise operators −



Operator	Description
& (bitwise and)	Binary AND Operator copies a bit to the result if it exists in both operands.
| (bitwise or)	Binary OR Operator copies a bit if it exists in either operand.
^ (bitwise XOR)	Binary XOR Operator copies the bit if it is set in one operand but not both.
~ (bitwise compliment)	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.
<< (left shift)	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.
>> (right shift)	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.
>>> (zero fill right shift)	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.

Relational Operator:

Relational Operators in Java are used to comparing two variables for equality, non-equality, greater than, less than, etc. Java relational operator always returns a boolean value – true or false.

Java has 6 relational operators.

1. == is the equality operator. This returns true if both the operands are referring to the same object, otherwise false.
2. != is for non-equality operator. It returns true if both the operands are referring to the different objects, otherwise false.
3. < is less than operator.
4. > is greater than operator.
5. <= is less than or equal to operator.
6. >= is greater than or equal to operator.

Boolean Logical Operator:

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.

These operators act on Boolean operands according to this table

A	B	A|B	A&B	A^B	!A
false	false	false	false	false	true
true	false	true	false	true	false
false	true	true	false	true	true
true	true	true	true	false	false

|	the OR operator
&	the AND operator
^	the XOR operator
!	the NOT operator
||	the short-circuit OR operator
&&	the short-circuit AND operator
==	the EQUAL TO operator
!=	the NOT EQUAL TO operator

Assignment Operators:

It's very common to see statement like the following, where you're adding something to a variable. Java Variables are assigned, or given, values using one of the assignment operators. The variable are always on the left-hand side of the assignment operator and the value to be assigned is always on the right-hand side of the assignment operator. The assignment operator is evaluated from **right to left**, so a = b = c = 0; would assign 0 to c, then c to b then b to a.

i = i + 2;
Here we say that we are assigning i's value to the new value which is i+2.

A shortcut way to write assignments like this is to use the += operator. It's one operator symbol so don't put blanks between the + and =.
i += 2; // Same as "i = i + 2"

Conditional Operators:

Java has the conditional operator. It's a ternary operator -- that is, it has three operands -- and it comes in two pieces, ? and :, that have to be used together. It takes the form
Boolean-expression ? **expression-1** : **expression-2**

The JVM tests the value of Boolean-expression. If the value is true, it evaluates expression-1; otherwise, it evaluates expression-2. For

Example

```
if (a > b) {

        max = a;

}

else {

        max = b;

}
```

Setting a single variable to one of two states based on a single condition is such a common use of if-else that a shortcut has been devised for it, the conditional operator, ?:. Using the conditional operator you can rewrite the above example in a single line like this:

```
max = (a > b) ? a : b;
```

Q.3 Discuss the conditional statement in Java.
Ans:

Java has the following conditional statements:

Use **if** to specify a block of code to be executed, if a specified condition is true

Use **else** to specify a block of code to be executed, if the same condition is false

Use **else if** to specify a new condition to test, if the first condition is false

Use **`switch`** to specify many alternative blocks of code to be executed

The if Statement :

Use the `if` statement to specify a block of Java code to be executed if a condition is `true`.

Syntax:

```
if (condition)
{
  // block of code to be executed if the condition is true
}
```

The else statement:

Use the `else` statement to specify a block of code to be executed if the condition is `false`.

Syntax :

```
if (condition)
 {
  // block of code to be executed if the condition is true
}
else
{
  // block of code to be executed if the condition is false
}
```

The else if statement:

Use the `else if` statement to specify a new condition if the first condition is `false`.

Syntax:

```
if (condition1)
{
  // block of code to be executed if condition1 is true
}
 else if (condition2)
{
  // block of code to be executed if the condition1 is false and condition2 is true
}
 else
{
  // block of code to be executed if the condition1 is false and condition2 is false
}
}
```

Switch Statement :

The java `switch` statement is used to select one of many code blocks to be executed.

Syntax:

```
switch(expression)
{
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

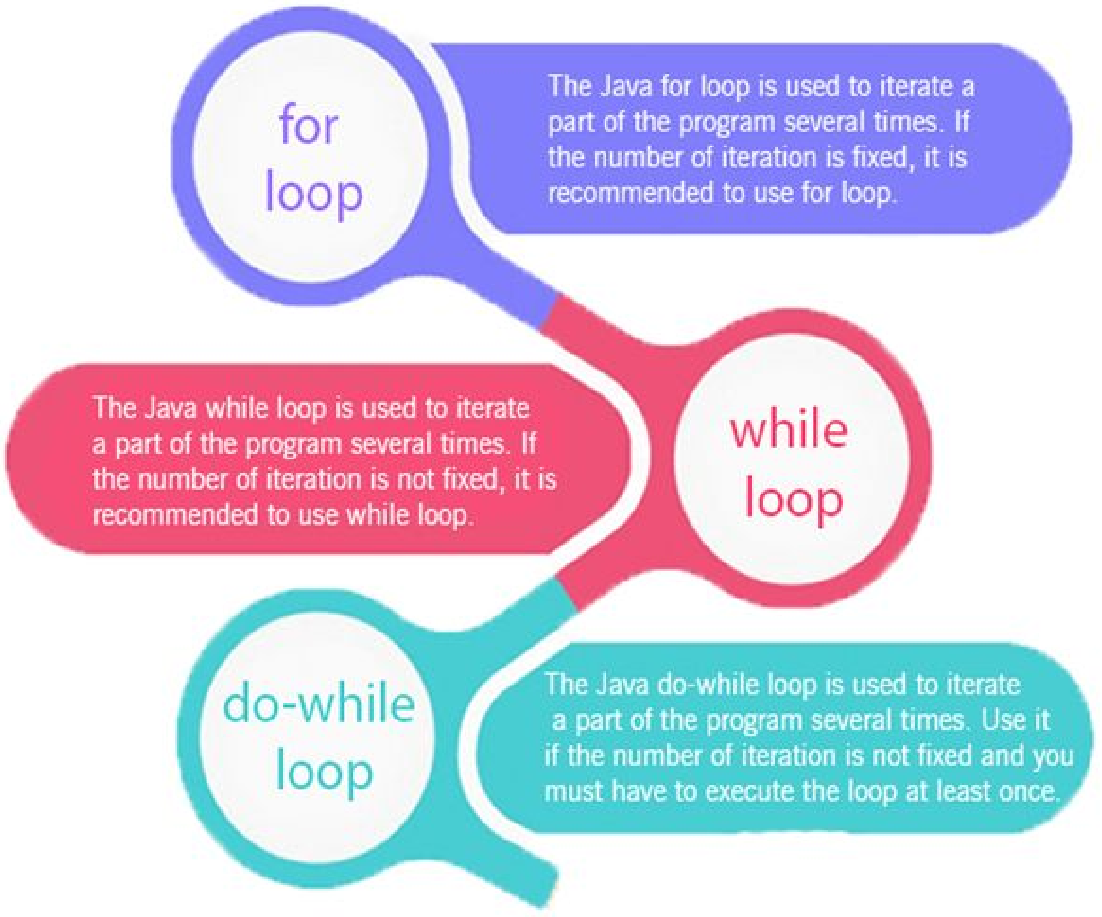
This is how it works:

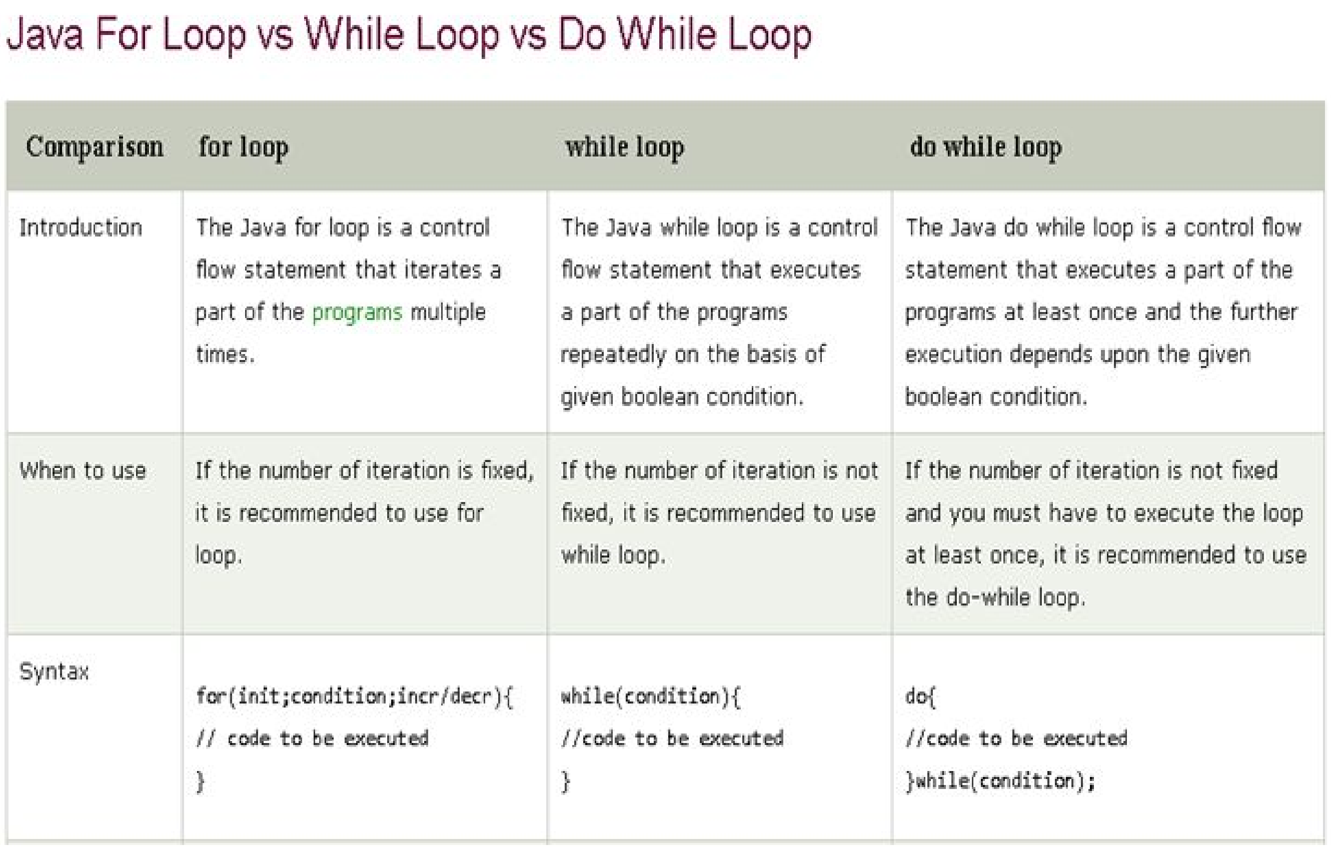
- The `switch` expression is evaluated once.
- The value of the expression is compared with the values of each `case`.
- If there is a match, the associated block of code is executed.
- The `break` and `default` keywords are optional, and will be described later in this chapter

Q.4.Discuss the Iterative Statement in Java.
Ans:

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true. There are three types of loops in Java.

- for loop
- while loop
- do-while loop


for loop
The Java for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.
while loop
The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.
do-while loop
The Java do-while loop is used to iterate a part of the program several times. Use it if the number of iteration is not fixed and you must have to execute the loop at least once.


Java For Loop vs While Loop vs Do While Loop

Comparison	for loop	while loop	do while loop
Introduction	The Java for loop is a control flow statement that iterates a part of the programs multiple times.	The Java while loop is a control flow statement that executes a part of the programs repeatedly on the basis of given boolean condition.	The Java do while loop is a control flow statement that executes a part of the programs at least once and the further execution depends upon the given boolean condition.
When to use	If the number of iteration is fixed, it is recommended to use for loop.	If the number of iteration is not fixed, it is recommended to use while loop.	If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use the do-while loop.
Syntax	`for(init;condition;incr/decr){` `// code to be executed` `}`	`while(condition){` `//code to be executed` `}`	`do{` `//code to be executed` `}while(condition);`

OR

Loops in Java

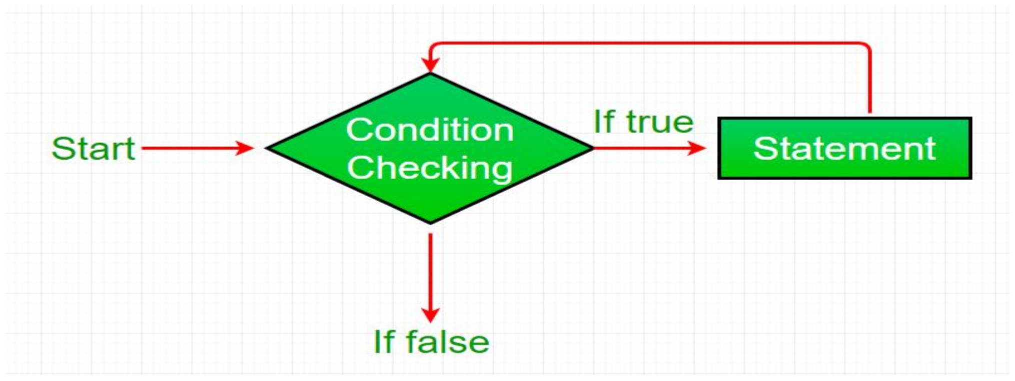
Looping in programming languages is a feature which facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true.
Java provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time.

I) while loop: A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.

Syntax :

```
while (boolean condition)
{
  loop statements...
}
```

Flowchart:



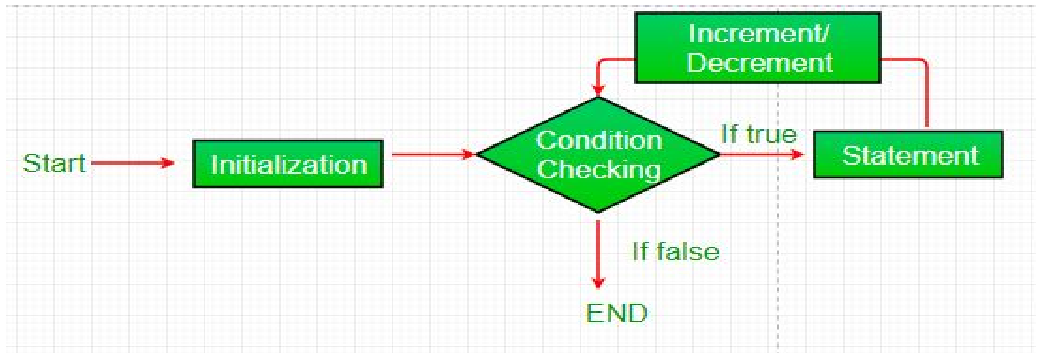
- **While loop starts with the checking of condition. If it evaluated to true, then the loop body statements are executed otherwise first statement following the loop is executed. For this reason it is also called Entry control loop**
- **Once the condition is evaluated to true, the statements in the loop body are executed. Normally the statements contain an update value for the variable being processed for the next iteration.**
- **When the condition becomes false, the loop terminates which marks the end of its life cycle.**

ii) for loop: for loop provides a concise way of writing the loop structure. Unlike a while loop, a for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.

Syntax:

```
for (initialization condition; testing condition; increment/decrement)
{
   statement(s)
}
```

Flowchart:



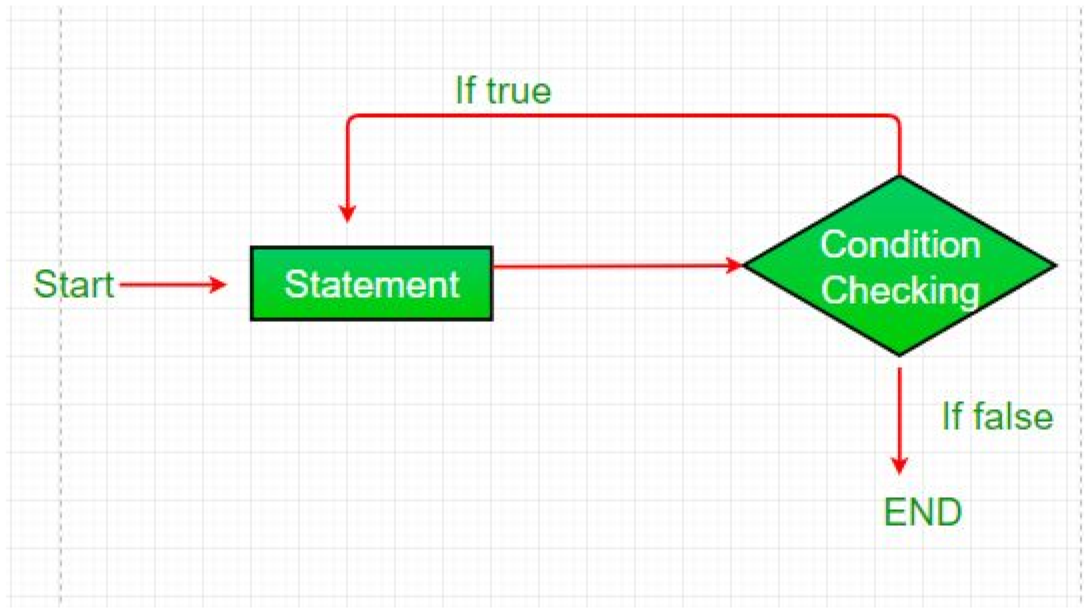
1. **Initialization condition: Here, we initialize the variable in use. It marks the start of a for loop. An already declared variable can be used or a variable can be declared, local to loop only.**
2. **Testing Condition: It is used for testing the exit condition for a loop. It must return a boolean value. It is also an Entry Control Loop as the condition is checked prior to the execution of the loop statements.**
3. **Statement execution: Once the condition is evaluated to true, the statements in the loop body are executed.**
4. **Increment/ Decrement: It is used for updating the variable for next iteration.**
5. **Loop termination:When the condition becomes false, the loop terminates marking the end of its life cycle.**

iii) do while: do while loop is similar to while loop with only difference that it checks for condition after executing the statements, and therefore is an example of Exit Control Loop.

Syntax:

```
do
{
   statements..
}
while (condition);
```

Flowchart:



1. do while loop starts with the execution of the statement(s). There is no checking of any condition for the first time.
2. After the execution of the statements, and update of the variable value, the condition is checked for true or false value. If it is evaluated to true, next iteration of loop starts.
3. When the condition becomes false, the loop terminates which marks the end of its life cycle.
4. It is important to note that the do-while loop will execute its statements atleast once before any condition is checked, and therefore is an example of exit control loop.

…………………………………….

Q.6. Discuss Jump Statements in Java.
Ans:
In Java, **Jump statements** are used to unconditionally transfer program control from one point to elsewhere in the program. *Jump statements* are primarily used to interrupt loop or switch-case instantly. Java supports three jump statements: **break, continue, and return.**

<u>Break Statement :</u>

The Java *break* statement is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

We can use Java break statement in all types of loops such as for loop, while loop and do-while loop.

Syntax:

```
jump-statement;
```

```
        break;
```

Example:

```
public class BreakExample
{
public static void main(String[] args)
{
   //using for loop
   for(int i=1;i<=10;i++)
        {
         if(i==5)
                {
                 //breaking the loop
                   break;
                 }
         System.out.println(i);
         }
}
}
```

Continue statement:

The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately. It can be used with for loop or while loop.

The Java *continue statement* is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition. In case of an inner loop, it continues the inner loop only.

We can use Java continue statement in all types of loops such as for loop, while loop and do-while loop.

Syntax:

```
jump-statement;

continue;
```

Example:

```
public class ContinueExample
{
public static void main(String[] args)
 {
   //for loop
   for(int i=1;i<=10;i++)
{
      if(i==5)
{
         //using continue statement
         continue;//it will skip the rest statement
      }
      System.out.println(i);
   }
}
}
```

THE Return Statement

This statement is mainly used in methods in order to terminate a method in between and return back to the caller method. It is an optional statement. That is, even if a method doesn't include a return statement, control returns back to the caller method after execution of the method. Return statement mayor may not return parameters to the caller method.

…………………………….

Q.9. Difference between break and continue Statement in java.

Ans:

break statement	continue statement
It terminates the execution of the remaining iteration of the loop.	It terminates only the current iteration of the loop.
break resumes the control of the program to the end of loop enclosing that 'break'.	continue resumes the control of the program to the next iteration of that loop enclosing continue.
It causes early termination of a loop.	It causes early execution of the next iteration.
break stops the continuation of the loop.	continuation of loop. continue do not stop the continuation of the loop, it only stops the current iteration.
the break can be used with switch, label.	continue can not be executed with switch and labels.