Operators and Control statements

What is Operator and its types?

An operator is a character that **represents an action**, for example + is an arithmetic operator that represents addition.

Types of Operator in Java

1) Basic Arithmetic Operators
2) Assignment Operators
3) Auto-increment and Auto-decrement Operators
4) Logical Operators
5) Comparison (relational) operators
6) Bitwise Operators
7) Ternary Operator

1) Basic Arithmetic Operators

Basic arithmetic operators are: +, -, *, /, %

+ is for addition.

– is for subtraction.

***** is for multiplication.

/ is for division.

% is for modulo.

Note: Modulo operator returns remainder, for example 10 % 5 would return 0

Example of Arithmetic Operators

```
public class ArithmeticOperatorDemo {

  public static void main(String args[]) {

    int num1 = 100;

    int num2 = 20;

   System.out.println("num1 + num2: " + (num1 + num2) );

    System.out.println("num1 - num2: " + (num1 - num2) );

    System.out.println("num1 * num2: " + (num1 * num2) );

    System.out.println("num1 / num2: " + (num1 / num2) );

    System.out.println("num1 % num2: " + (num1 % num2) );

  }

}
```

OUTPUT:
num1 + num2: 120
num1 - num2: 80
num1 * num2: 2000
num1 / num2: 5
num1 % num2: 0

2) Assignment Operators

Assignments operators in java are: =, +=, -=, *=, /=, %=
num2 = num1 would assign value of variable num1 to the variable.

num2+=num1 is equal to num2 = num2+num1

num2-=num1 is equal to num2 = num2-num1

num2*=num1 is equal to num2 = num2*num1

num2/=num1 is equal to num2 = num2/num1

num2%=num1 is equal to num2 = num1

Example of Assignment Operators

```
public class AssignmentOperatorDemo {

  public static void main(String args[]) {

    int num1 = 10;

    int num2 = 20;

    num2 = num1;//num2=10

    System.out.println("= Output: "+num2);

    num2 += num1;//num2=10///num2=num2+num1=20

    System.out.println("+= Output: "+num2);

    num2 -= num1;//num2=num2-num1//num2=20

    System.out.println("-= Output: "+num2);

    num2 *= num1;

    System.out.println("*= Output: "+num2);

    num2 /= num1;

    System.out.println("/= Output: "+num2);

    num2 %= num1;

    System.out.println("%= Output: "+num2);

  }

}
```

Output:

= Output: 10
+= Output: 20
-= Output: 10
*= Output: 100
/= Output: 10
%= Output: 0

3) Auto-increment and Auto-decrement Operators

++ and —

num++ is equivalent to num=num+1;

num–- is equivalent to num=num-1;

Example of Auto-increment and Auto-decrement Operators

```
public class AutoOperatorDemo {
  public static void main(String args[]){
    int num1=100;
    int num2=200;
    num1++;
    num2--;
    System.out.println("num1++ is: "+num1);
    System.out.println("num2-- is: "+num2);
  }
}
```

Output:

num1++ is: 101
num2-- is: 199

4) Logical Operators

Logical Operators are used with binary variables. They are mainly used in conditional statements and loops for evaluating a condition.

Logical operators in java are: &&, ||, !

Let’s say we have two boolean variables b1 and b2.

b1&&b2 will return true if both b1 and b2 are true else it would return false.

b1||b2 will return false if both b1 and b2 are false else it would return true.

!b1 would return the opposite of b1, that means it would be true if b1 is false and it would return false if b1 is true.

Example of Logical Operators

```
public class LogicalOperatorDemo {
   public static void main(String args[]) {
      boolean b1 = true;//1
      boolean b2 = false;//0

      System.out.println("b1 && b2: " + (b1&&b2));
      System.out.println("b1 || b2: " + (b1||b2));
      System.out.println("!(b1 && b2): " + !(b1&&b2));
   }
}
```

Output:

b1 && b2: false
b1 || b2: true
!(b1 && b2): true

5) Comparison(Relational) operators

We have six relational operators in Java: ==, !=, >, <, >=, <=

== returns true if both the left side and right side are equal

!= returns true if left side is not equal to the right side of operator.

> returns true if left side is greater than right.

< returns true if left side is less than right side.

>= returns true if left side is greater than or equal to right side.eg j=0;j<10,j++

<= returns true if left side is less than or equal to right side.

6) Bitwise Operators

There are six bitwise Operators: &(AND), |(OR), ^(XOR), ~(BITWISE COMPLIMENT), <<, >>

num1 = 11; /* equal to 00001011*/ 128 64 32 16 8421
num2 = 22; /* equal to 00010110 */

a = 0011 1100 11110000 0000 1111

---

b = 0000 1101

---

0 0 1 1 0 0 01

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

<< (left shift)

Binary Left Shift Operator <<:. The left operands value is moved left by the number of bits specified by the right operand.

A << 2 will give 240 which is 1111 0000

>> (right shift)

Binary Right Shift Operator >>. The left operands value is moved right by the number of bits specified by the right operand.

A >> 2 will give 15 which is 1111

7) Ternary Operator

This operator evaluates a boolean expression and assign the value based on the result.

Syntax:

variable num1 = (expression) ? value if true : value if false

If the expression results true then the first value before the colon (:) is assigned to the variable num1 else the second value is assigned to the num1.

Example of Ternary Operator

```
public class TernaryOperatorDemo {

  public static void main(String args[]) {

    int num1, num2;

    num1 = 25;

    /* num1 is not equal to 10 that's why
     * the second value after colon is assigned
     * to the variable num2
     */

    num2 = (num1== 10) ? 100: 200;

    System.out.println( "num2: "+num2);

    /* num1 is equal to 25 that's why
     * the first value is assigned
     * to the variable num2
     */

    num2 = (num1 == 25) ? 100: 200;

    System.out.println( "num2: "+num2);

  }

}
```

Output:
num2: 200
num2: 100s

Control Statements

If Else Statement

In Java, if statement is used for testing the conditions. The condition matches the statement it returns true else it returns false. There are four types of If statement they are:

if statement

if-else statement

if-else-if ladder

nested if statement

If –Statement

if Statement

In Java, if statement is used for testing conditions. It is used for only true condition.

Syntax:

```
if(condition)
{
//code
}
```

```
public class IfDemo1 {
public static void main(String[] args)
{
        int marks=70;
        if(marks > 65)
                {
                System.out.print("First division");
                }
        }
}
```

If-else statement

if-else Statement

In Java, the if-else statement is used for testing conditions. It is used for true as well as for false condition.

Syntax:

```
if(condition)
{
//code for true
}
Else
{
//code for false
}
```

```
public class IfElseDemo1 {
public static void main(String[] args)
{
int marks=50;
if(marks > 65)
{
System.out.print("First division");
}
else
{
System.out.print("Second division");
}
}
}
```

if-else-if ladder Statement

In Java, the if-else-if ladder statement is used for testing conditions. It is used for testing one condition from multiple statements.

Syntax:

```
if(condition1)
{
//code for if condition1 is true
}
else if(condition2)
{
//code for if condition2 is true
}
else if(condition3)
{
//code for if condition3 is true
}
...
else
{
//code for all the false conditions
}
```

Example

```
public class IfElseIfDemo1 {
public static void main(String[] args) {
int marks=75;
   if(marks<50){
System.out.println("fail");
   }
   else if(marks>=50 && marks<60){ //50-59
System.out.println("D grade");
   }
   else if(marks>=60 && marks<70){ //60-69
System.out.println("C grade");
   }
 else if(marks>=70 && marks<80){ //70-79
System.out.println("B grade");
   }
   else if(marks>=80 && marks<90){
System.out.println("A grade");
}else if(marks>=90 && marks<100){
System.out.println("A+ grade");
}else{
System.out.println("Invalid!");
   }
}
}
```

Nested if statement

In Java, the Nested if statement is used for testing conditions. In this, one if block is created inside another if block when the outer block is true then only the inner block is executed.

Syntax:

```
if(condition)
{
        //statement
    if(condition)
{
        //statement
   }
}
```

```
public class NestedIfDemo1 {
public static void main(String[] args)
{
int age=25;
int weight=70;
if(age>=18)
{
if(weight>50)
{
System.out.println("You are eligible");
}
}
}
}
```

For Loop

In Java, for loop is used for executing a part of the program again and again. When the number of execution is fixed then it is suggested to use for loop. In java there are 3 types of for loops, they are as follows:

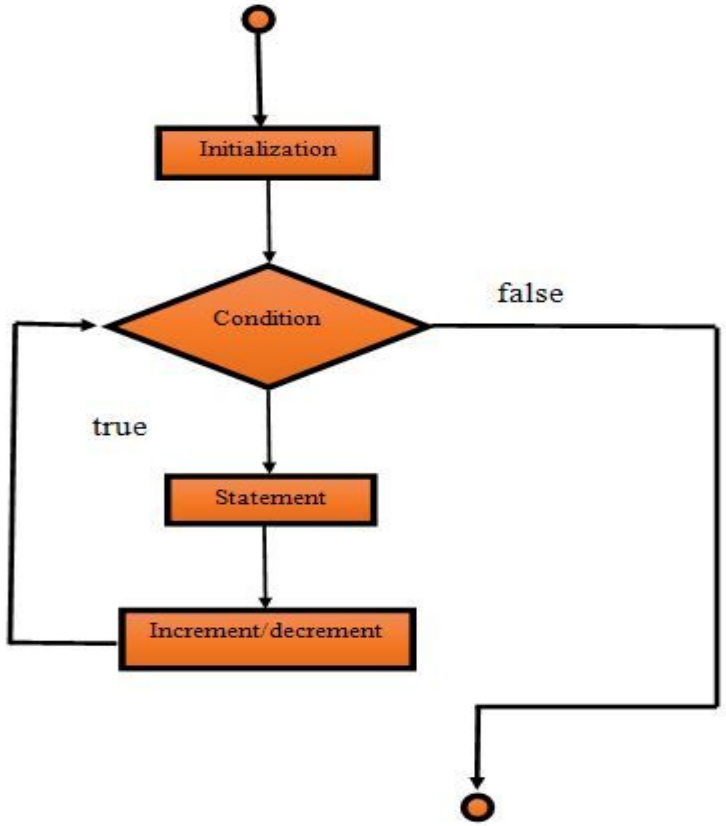
- 1.Simple for loop

2.For-each loop

3.labelled for loop

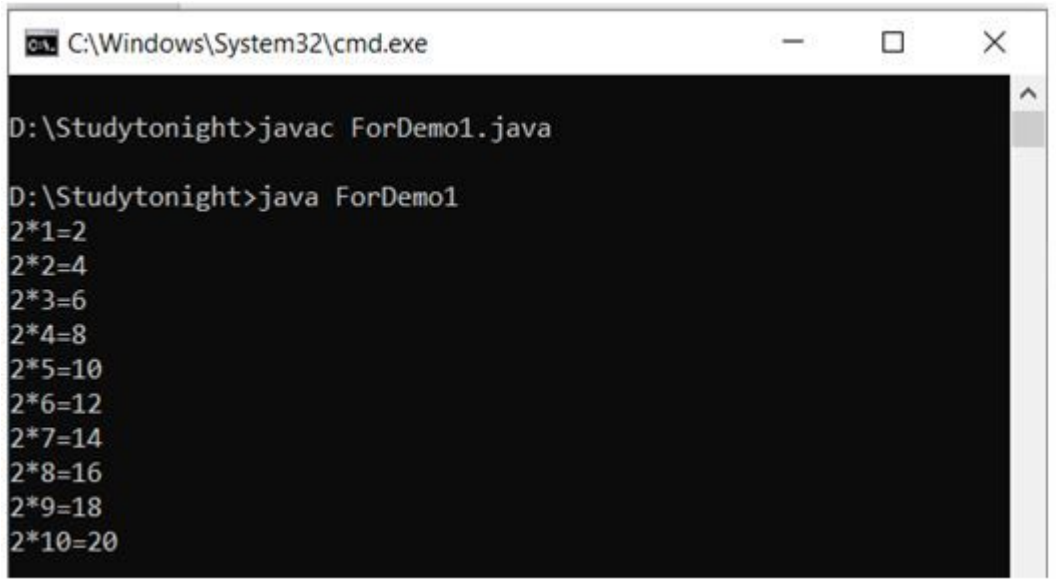
Syntax:

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



Example

```
public class ForDemo1
{
public static void main(String[] args)
	{
	int n, i;
	n=2;
	for(i=1;i<=10;i++)
	{
	System.out.println(n+"*"+i+"="+n*i);
	}
}
}
```



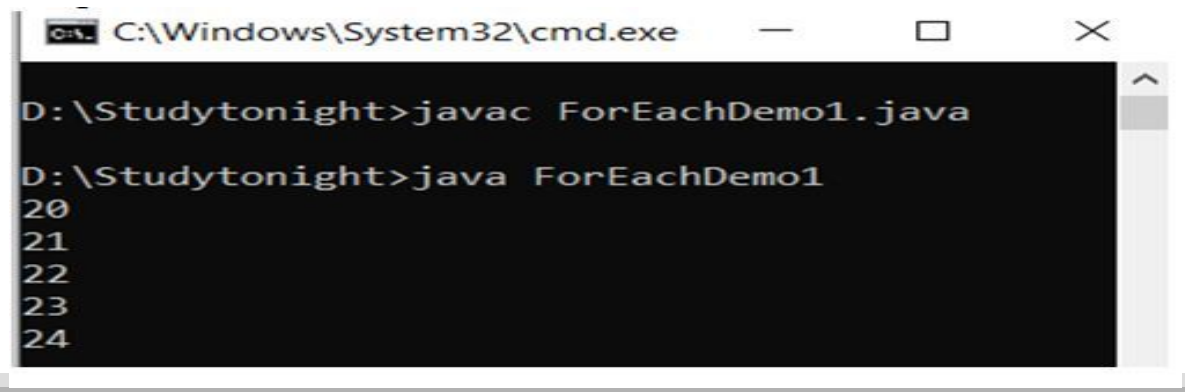
for-each Loop

In Java, for each loop is used for traversing array or collection. In this loop, there is no need for increment or decrement operator.

Syntax:

```
for(Type var:array)
{
//code for execution
}
```

```
public class ForEachDemo1
{
public static void main(String[] args)
        {
                int a[]={20,21,22,23,24};
                for(int i:a)
{
                        System.out.println(i);
                }
}
}
```



Labelled For Loop

In Java, Labelled For Loop is used to give label before any for loop. It is very useful for nesting for loop.

Syntax:

labelname:

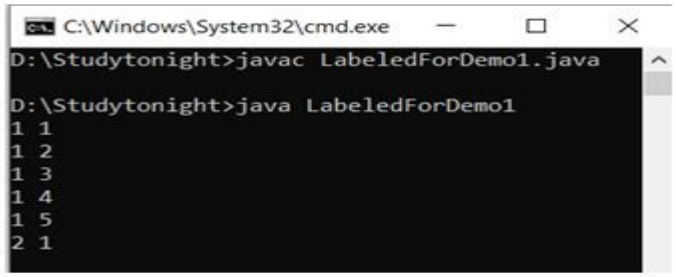
for(initialization;condition;incr/decr)

{

//code for execution

}

```
public class LabeledForDemo1
{
public static void main(String[] args)
{
        num:
                for(int i=1;i<=5;i++)
{
        num1:
                for(int j=1;j<=5;j++)
{
                        if(i==2&&j==2)
{
                        break num;
                        }
                System.out.println(i+" "+j);
                }
                }
}
}
```



Switch Statement

In Java, the switch statement is used for executing one statement from multiple conditions. it is similar to an if-else-if ladder. In a switch statement,the expression can be of byte, short, char and int data types. From JDK7 enum, String class and the Wrapper classes can also be used. Following are some of the rules while using the switch statement:

There can be one or N numbers of cases.

The values in the case must be unique.

Each statement of the case can have a break statement. It is optional.

Syntax

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

While Loop

In Java, While loop is a control statement. It is used for iterating a part of the program several times. When the number of iteration is not fixed then while loop is used.

Syntax:

```
while(condition)
{
//code for execution
}
```

do-while loop

In Java, the do-while loop is used to execute a part of the program again and again. If the number of iteration is not fixed then the do-while loop is used. This loop executes at least once because the loop is executed before the condition is checked.

Syntax:

```
do
{
//code for execution
}
while(condition);
```

Break and Continue Statement

In Java, a break statement is used inside a loop. The loop is terminated immediately when a break statement is encountered and resumes from the next statement.

Syntax:

```
jump-statement;

break;
```

Eg:

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  System.out.println(i);
}
```

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Eg:

```
for (int i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  System.out.println(i);
}
```