- **Name of Teacher:** Miss Nita N. Bargale
- **Class:** B.Sc. Computer Science (Entire)- III
- **Course Title:** Java programming

Control statements in java

The control statements are used to control the flow of execution of the program

This execution order depends

on the supplied data values and the conditional logic.

Control Statements can be divided into three categories, namely

1. Selection statements/**Decision-Making Statements**
2. Iteration statements/**Looping Statements**
3. Jump statements/**Branching Statements**

1. Decision-Making Statements

statements that determine which statement to execute and when are known as decision-making statements. The flow of the execution of the program is controlled by the control flow statement.

Following decision-making statements available in java.

- If statement
- If-else statement
- Switch statement

if Statement

In Java, "if" is a conditional statement. It will provide execution of one of two statements (or blocks), depending on the condition.

Syntax

```
if(condition)
{
   statements;
   ...
   ...
}
```

If the condition is true then the statements inside the if block will be executed. If the condition is false then the statements inside the if block will not be executed and the execution will start from the statements that are next to the if block.

- **Example**

```
import java.util.Scanner;
public class Test
{
   public static void main(String args[])
   {
      int b,c;
      Scanner s = new Scanner(System.in);
      System.out.print("b is : ");
      b=scnr.nextInt();
      System.out.print("c is : ");
      c=s.nextInt();
      if (b > c)
      {
         System.out.println("b is greater than c");
      }
       System.out.println("example for the comparison of two numbers");
     }
```

- If-else Statement

If the condition is true then the statements inside the if block will be executed and if the condition is false then the statements inside the else block will be executed.

Syntax

```
if (condition)
{
   statement;
}
else
{
   statement;
}
```

```
import java.util.Scanner;
public class Demo
{
   public static void main(String args[])
   {
      int a,b;
      Scanner scnr = new Scanner(System.in);
      System.out.println("a is : ");
      a=scnr.nextInt();
      System.out.println("b is : ");
      b=scnr.nextInt();
      if (a > b)
      {
         System.out.println("a is largest");
      }
      else
      {
         System.out.println("b is largest");
      }
   }
}
```

Switch Statement

when there are several options and we have to choose only one option from them,we can use switch statement.depending upon the selected option,a perticular task is performed. we can only use int, char, byte and short data types.

Syntax

```
switch (expression)
{
   case 1:
   {
      statement;
   }
   break;
   case 2:
   {
      statement;
   }
   break;
   .
   .
   case N:
   {
      statement;
   }
   break;
   default:
   {
      statement;
   }
   break;
}
```

```
import java.util.Scanner;
public class Demo
{
   public static void main(String[] args)
   {
      int x,y,r;
      double z;
      Scanner scnr = new Scanner(System.in);
      System.out.print("Enter x : ");
      x=scnr.nextInt();
      System.out.print("Enter y : ");
      y=scnr.nextInt();
      System.out.println("1 : Addition");
      System.out.println("2 : Subtraction");
      System.out.println("3 : Multiplication");
      System.out.println("4 : Division");
```

```
System.out.print("Requirement : ");
        r=scnr.nextInt();
        switch(r)
        {
            case 1:
            {
                z=x+y;
                System.out.println(z);
            }
            break;

}
```

```
    case 2:
      {
         z=x-y;
         System.out.println(z);
      }
      break;
      case 3:
      {
         z=x*y;
         System.out.println(z);
      }
      break;
      case 4:
      {
         z=x/y;
         System.out.println(z);
      }
      break;
      default:
      {
         System.out.println("Requirement is invalid");
      }
   }
}
```

A switch statement in java is used to execute a single statement from multiple conditions.
Certain points must be noted while using the switch statement:

One or N number of case values can be specified for a switch expression.

Case values that are duplicate are not permissible. A compile-time error is generated by the compiler if unique values are not used.

The case value must be literal or constant. Variables are not permissible.

Usage of break statement is made to terminate the statement sequence. It is optional to use this statement. If this statement is not specified, the next case is executed.

1. Write a prog to print given no is even
2. Write a prog to print given no is even or not
3. Write a prog to find largest among 3 numbers
4. Write a prog to test inputted no is positive or negative
5. Write a prog to accept 3 nos from user representing the sides of the triangles. Find out if the triangle is equilateral or not.
6. Write a prog to display days of week
7. Write a prog to display month of year
8. Write a prog to display no of days in a month (input –month no and year)

- **Looping Statements/iteration statement**

  Statements that execute a block of code repeatedly until a specified condition is met are known as looping statements. when this condition is false ,the iteration terminates. Following the types of loops
- while loop
- do-while loop
- for loop
- for -each

- While loop
- This looping structure is also called as pre-tested looping structure or entry –controlled loop. This statement repeats the execution of a set of statements while the condition is TRUE. once the condition becomes false the loop gets terminated
- There must be increment/decrement statement inside the while block that should make the evaluation result to return false ,otherwise an infinite loop is created
- **Syntax:**

```
while ( Test Condition)
{
 Statement;
}
```

```
public class Example1 {
public static void main(String args[] )
{
   int i=1;
   while(i<=10)
{
      System.out.println(i);
   i++;
   }
}
}
```

- **do..**while
- The do-while loop is similar to the while loop, the only difference being that the condition in the do-while loop is evaluated after the execution of the loop body. This guarantees that the loop is executed at least once.
- This loop is also call exit–control loop
- **Syntax:**

```
do
{
  Body of loop
}while(condition);
```

```
Class Demo1
 {
public static void main(String args [])
 {
Int i=1;
do
{
 System.out.println(i);
 i++;
 }while(i<=10);
 }
 }
```

- **For loop**
- It executes the code until condition is false.
- It is used when number of iterations are known.
- Syntax:
- for(initialization; condition; increment/decrement)
- {
  //Body of loop
  }

```
class forLoop
{
public static void main(String args[])
{
for (int i = 5; i <= 10; i++)
System.out.println(i);
}
}
```

- for-each loop
- For-Each loop is used to traverse through elements in an array
- for-each loop is the simpler form of the usual ‘for’ loop, usually used to iterate over an array or collection object.
- Class Demo{
- public static void main(String args[]){
- int a[] = {10,15,20,25,30};
- for (int i : a) {
- System.out.println(i);
- }
- }