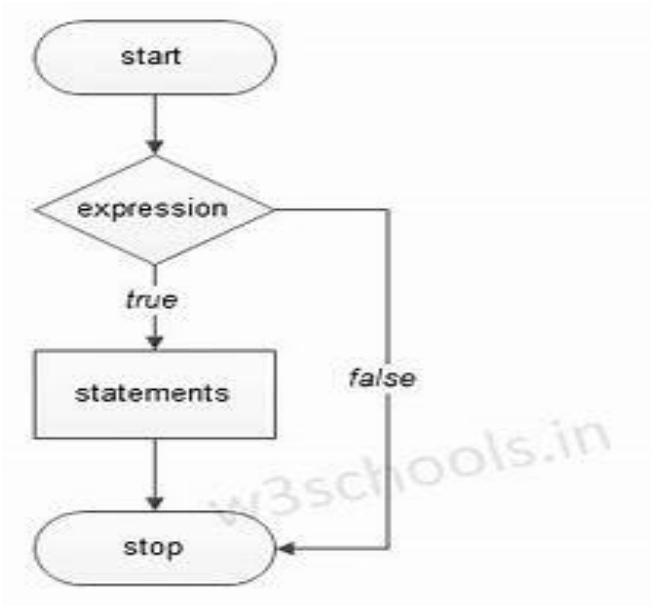
Week – 05

Conditional and Iterative statements

JAVA'S SELECTION STATEMENTS [Decision Making, Branching]

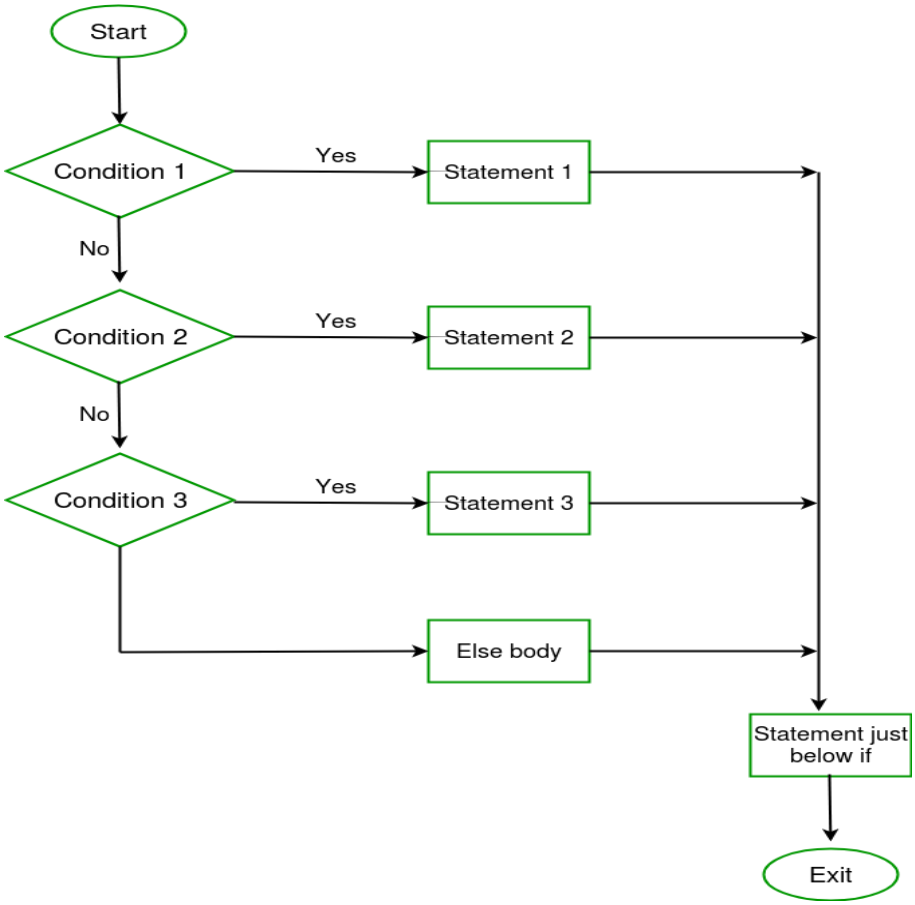
Java supports two selection statements: if and switch. These statements allow you to control the flow of your program's execution based upon conditions known only during run time.



Simple if

if (*condition*)
statement1;
else
statement2;

.



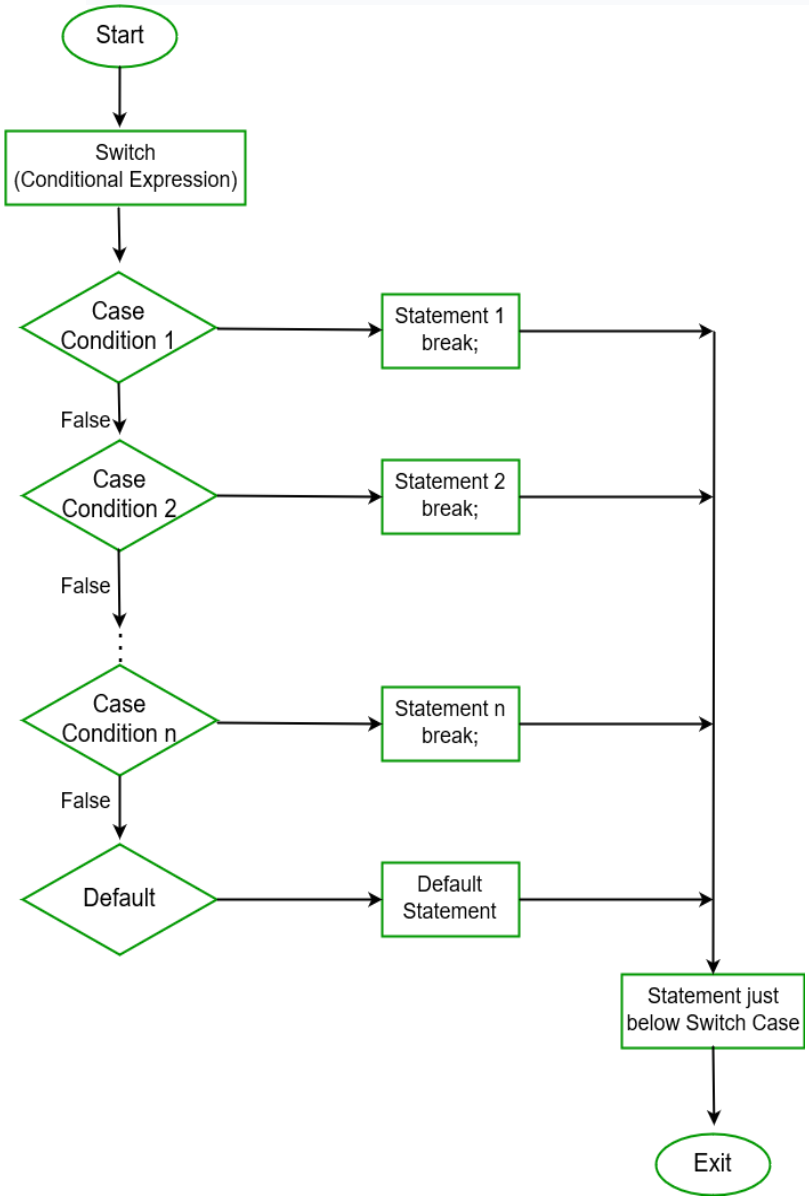
The if-else-if Ladder
if(*condition*)
statement;
else i*f(condition*)
statement;
else if(*condition*)
statement;

.
.
else
statement;

Switch Statement

The expression is evaluated once and compared with the values of each **case label**.

- If there is a match, the corresponding code after the matching **case label** is executed. For example, if the value of the expression is equal to value2, the code after case value2: is executed.
- If there is no match, the code after default: is executed.



switch (*expression*)
{
case *value1*: // statement sequence
break;
case *value2*: // statement sequence
break;
:
:
:
case *valueN*: // statement sequence
break;
default: // default statement sequence
}

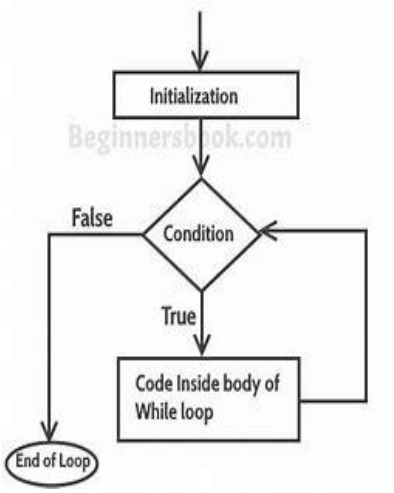
1.27 ITERATION STATEMENTS [Decision Making, Looping]

Java's iteration statements are for, while, and do-while. These statements create what we commonly call loops. As you probably know, a loop repeatedly executes the same set of instructions until a termination condition is met.

While

It repeats a statement or Block while its controlling expression is true.

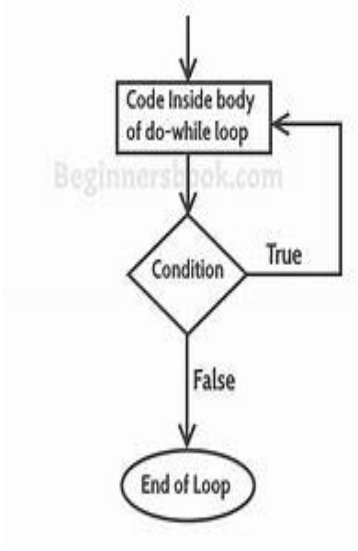
while(condition)
{
// body of loop
}



do-while

Sometimes it is desirable to execute the body of a while loop at least once, even if the conditional expression is false to begin with

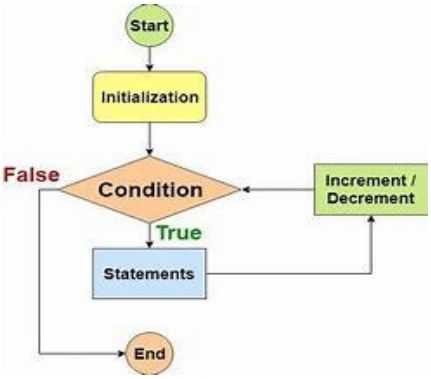
do {
// body of loop
} while (condition);



for

Here is the general form of the **for** statement:

for(*initialization*; *condition*; *iteration*)
{
// body
}



1.28 JUMP STATEMENTS [Labelled Loops]

Java supports three jump statements: break, continue, and return. These statements transfer control to another part of your program. Each is examined here.

In addition to the jump statements discussed here, Java supports one other way that you can change your program's flow of execution: through exception handling. Exception handling provides a structured method by which run-time errors can be trapped and handled by your program. It is supported by the keywords try, catch, throw, throws, and finally. In essence, the exception handling mechanism allows your program to perform a nonlocal branch.

Break

In Java, the **break** statement has three uses.
First, as you have seen, it terminates a statement sequence in a **switch** statement.

Second, it can be used to exit a loop.

Third, it can be used as a "civilized" form of goto.

continue

Sometimes it is useful to force an early iteration of a loop. That is, you might want to continue running the loop, but stop processing the remainder of the code in its body for this particular iteration.

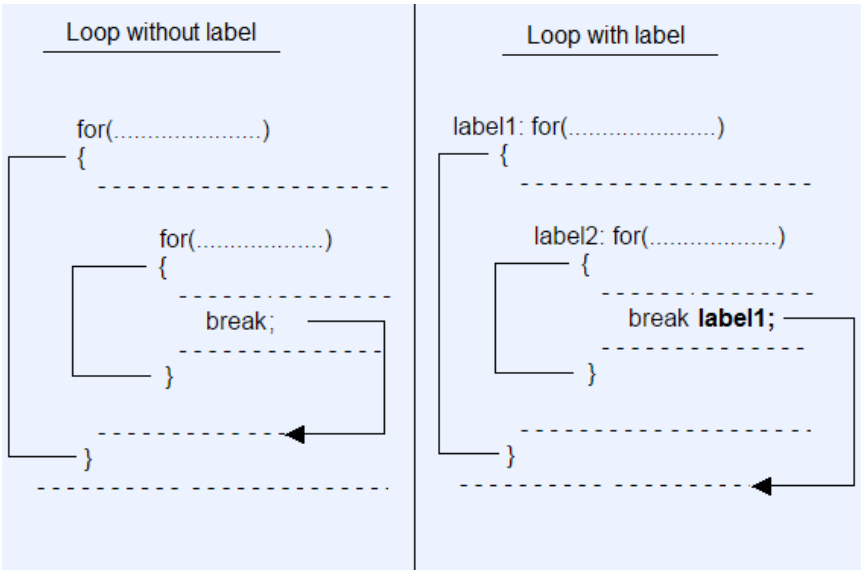
For all three loops (while, do-while, for), any Intermediate code is bypassed.

return

The last control statement is **return**. The **return** statement is used to explicitly return from a method. That is, it causes program control to transfer back to the caller of the method.

Thus, the **return** statement immediately terminates the method in which it is executed.

we use label before the for loop. It is useful if we have nested for loop so that we can break/continue specific for loop.



Exercise 1: Get a number from the user and print whether it is positive or negative number

```
import java.util.Scanner;
public class Exercise1 {


  public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        System.out.print("Input number: ");
        int input = in.nextInt();

        if (input > 0)
        {
            System.out.println("Number is positive");
        }
        else if (input < 0)
        {
            System.out.println("Number is negative");
        }
        else
        {
            System.out.println("Number is zero");
        }
    }
```

Exercise 2: Find the greatest of three numbers

```
public class Exercise2 {
        public static void main(String[] args) {
        int num1=30; int num2=50; int num3=60;

if (num1 > num2)
    if (num1 > num3)
       System.out.println("The greatest: " + num1);

   if (num2 > num1)
     if (num2 > num3)
        System.out.println("The greatest: " + num2);

     if (num3 > num1)
      if (num3 > num2)
       System.out.println("The greatest: " + num3);
               }
        }
```

Exercise 3: Demo of if else ladder

```
import java.util.Scanner;
public class Exercise3 {
public static void main(String args[])
{
     Scanner in = new Scanner(System.in);
      System.out.print("Input sem: ");
       int sem = in.nextInt();

if(sem==1)
    System.out.println("You are 1st sem");
else if(sem==2)
   System.out.println("You are 2nd sem");
else if(sem==3)
     System.out.println("You are 3rd sem");
else if(sem==4)
     System.out.println("You are 4th sem");
else if(sem==5)
     System.out.println("You are 5th sem");
else if(sem==6)
     System.out.println("You are 6th sem");
else
 System.out.println("Please give the valid sem.");

 }
```

Exercise 4: Demo of Switch statement

```
mport java.util.Scanner;
public class Exercise4 {
       public static void main(String args[])
       {

           Scanner in = new Scanner(System.in);
           System.out.print("Input sem: ");
           int sem = in.nextInt();

            switch (sem)
            {
              case (1):
                System.out.println("You are 1st sem");
                    break;
              case (2):
                System.out.println("You are 2nd sem");
                    break;
              case (3):
                System.out.println("You are 3rd sem");
                   break;
              case (4):
                System.out.println("You are 4th sem");
                   break;
              case (5):
                System.out.println("You are 5th sem");
                    break;
              case (6):
                System.out.println("You are 6th sem");
                    break;
     default:
    System.out.println("Please give the valid sem.");
                    break;
            }
        }
}
```

Exercise 5: Addition of two matrices using for loops

```
public class Exercise5
{
public static void main(String args[])
{
//creating two matrices
int a[][]={{1,3,4},{2,4,3},{3,4,5}};
int b[][]={{1,3,4},{2,4,3},{1,2,4}};

int c[][]=new int[3][3];

//adding and printing addition of 2 matrices
for(int i=0;i<3;i++)
{
  for(int j=0;j<3;j++){
      c[i][j]=a[i][j]+b[i][j];
       System.out.print(c[i][j]+"\t");
}
System.out.println();//new line
}
}
```

Exercise 6: Addition of two matrices using while loops

```
public class Exercise6
{
public static void main(String args[])
{
//creating two matrices
int a[][]={{1,3,4},{2,4,3},{3,4,5}};
int b[][]={{1,3,4},{2,4,3},{1,2,4}};

int c[][]=new int[3][3];
int i=0; int j;
while(i<3)
{
       j=0;
       while(j<3)
       {
              c[i][j]=a[i][j]+b[i][j];
              System.out.print(c[i][j]+"\t");
              j++;

       }
       System.out.println();//new line
       i++;
}
}
```

Exercise 7: Addition of two matrices using do while loops

```
public class Exercise7
{
public static void main(String args[])
{
//creating two matrices
int a[][]={{1,3,4},{2,4,3},{3,4,5}};
int b[][]={{1,3,4},{2,4,3},{1,2,4}};

int c[][]=new int[3][3];
int i=0; int j;
do
{
       j=0;
       do
       {
              c[i][j]=a[i][j]+b[i][j];
              System.out.print(c[i][j]+"\t");
              j++;

       }while(j<3);
       System.out.println();//new line
       i++;
}while(i<3);
}
}
```

Exercise 8: Demo of break statement with for loop

```
public class Exercise8
{
  public static void main(String args[])
  {
        for (int i = 0; i < 10; i++)
       {
              if (i == 4)
              {
                break;
              }
              System.out.println(i);
            }
  }
```

Exercise 9: Demo of continue statement with for loop

```
public class Exercise9
{
  public static void main(String args[])
  {
        for (int i = 0; i < 10; i++)
       {
              if (i == 4)
              {
                continue;
              }
              System.out.println(i);
            }
  }
```

Exercise 10: Demo of break statement with while loop

```
public class Exercise10
{
  public static void main(String args[])
  {
    int i = 0;
  while (i < 10) {
        System.out.println(i);
        i++;
        if (i == 4) {
          break;
        }
       }
  }
}
```

Exercise11: Demo of continue statement with while loop

```
public class Exercise11
{
  public static void main(String args[])
  {
int i = 0;
  while (i < 10) {

              if (i == 4) {
                    i++;
          continue;
        }
        System.out.println(i);
        i++;
       }
  }
}
```