Marwadi education foundation

Unit – 2
Selections , Mathematical functions and loops

Prepared By
Prof. Ravikumar R N
Assistant Professor, CE Dept.

KNOWLEDGE IS THE CURRENCY
FOR THE 21st CENTURY

Selections

1. If Statement
2. Two way if-else statement
3. Nested if and multi-way if else statement
4. Switch statement :

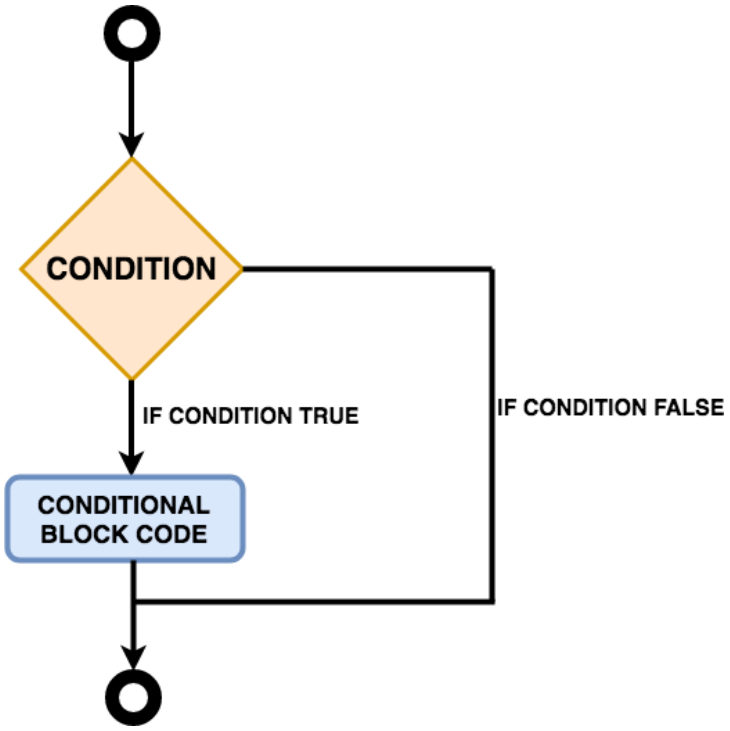
```
switch (switch-expression) {
    case value1: statement(s);
                 break;
    ..... default : statement;
}
```

5. Conditional Expression (Ternary) :

```
boolean-expression ? expression1 : expression2;
```

If Statement

The Java if statement tests the condition. It executes the if block if condition is true.



Syntax:

```
if(condition){
//code to be executed
}
```

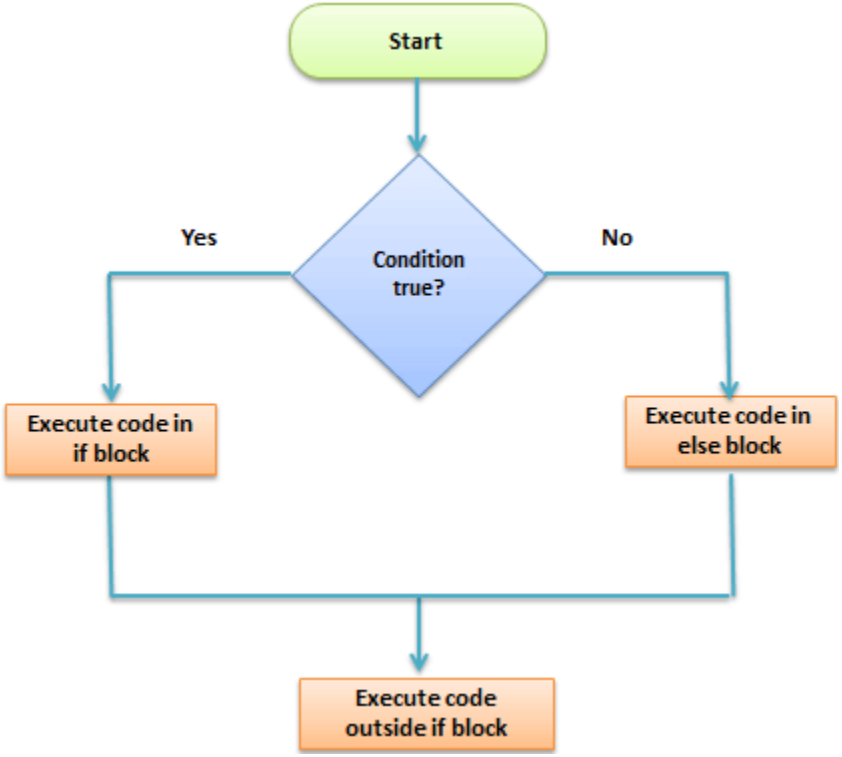
Check Program: Unit – 2 → IfDemo.java

Two way If (if.. else Statement)

The Java if-else statement also tests the condition. It executes the if block if condition is true otherwise else block is executed.

Syntax:

```
if(condition){
  //code if condition is true
  }else{
  //code if condition is false
  }
```



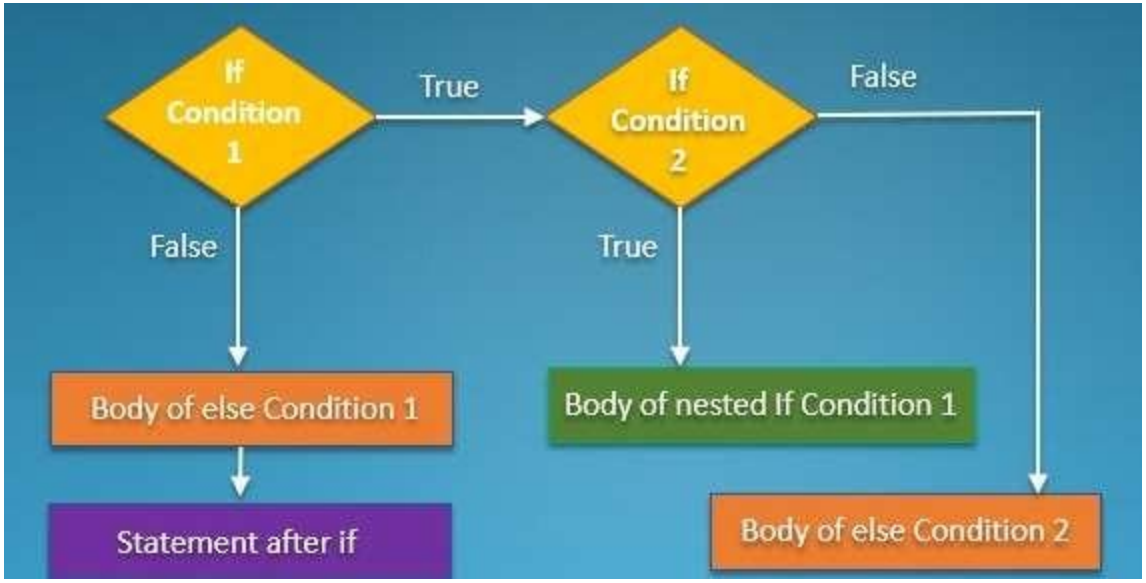
Check Program: Unit – 2 → IfElseDemo.java

Nested if statement

The nested if statement represents the if block within another if block. Here, the inner if block condition executes only when outer if block condition is true.

Syntax:

```
if(condition){
    //code to be executed
        if(condition){
            //code to be executed
    }
}
```



Check Program: Unit – 2 → NestedIfDemo.java

Multi way if... ladder if

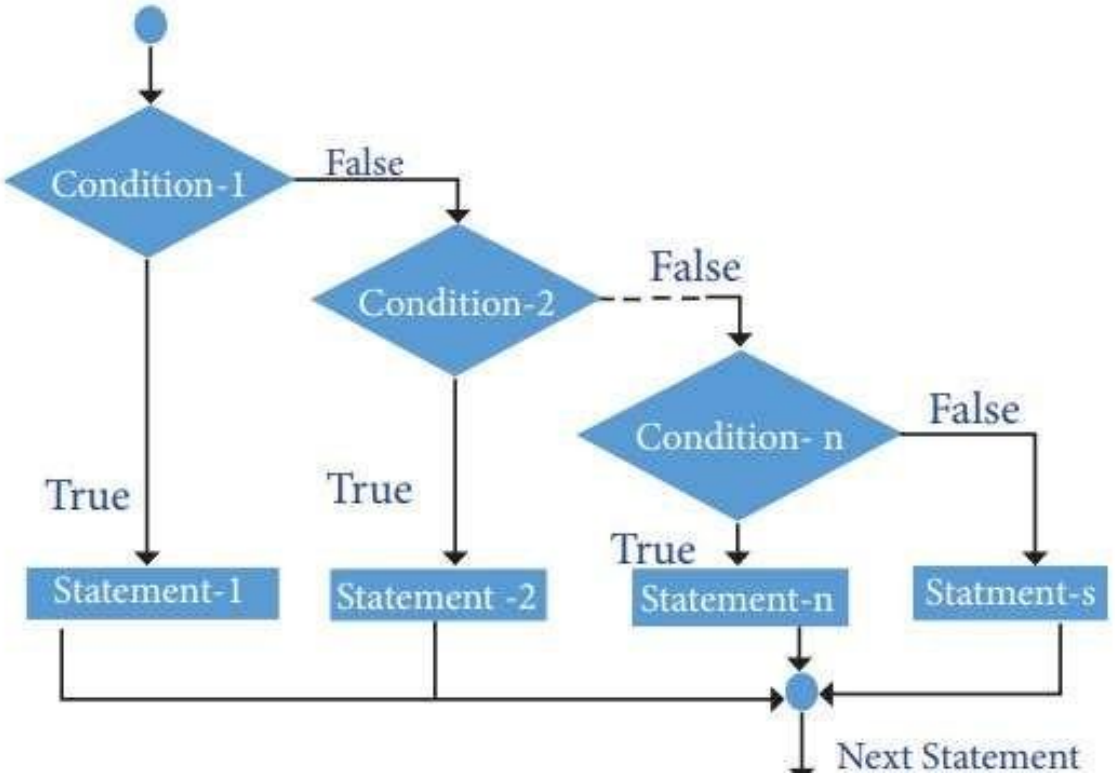
The if-else-if ladder statement executes one condition from multiple statements.

Syntax:

```
if(condition1){
//code to be executed if condition1 is true
}else if(condition2){
//code to be executed if condition2 is true
}
else if(condition3){
//code to be executed if condition3 is true
}
...
else{
//code to be executed if all the conditions are false
}
```

Multi way if... ladder if

The if-else-if ladder statement executes one condition from multiple statements.



Check Program: Unit – 2 → LadderIfDemo.java

Multi way if… ladder if

```
class IfElseLadder{
public static void main(String[] args)
{
        // initializing expression
        int i = 20;

        // condition 1
        if (i == 10)

System.out.println("i is 10\n");

        // condition 2
        else if (i == 15)

System.out.println("i is 15\n");

// condition 3
else if (i == 20)
System.out.println("i is 20\n");
else
System.out.println("i is not present\n");

System.out.println("Outside if-else-if");
        }
}
```

Java Switch Statement

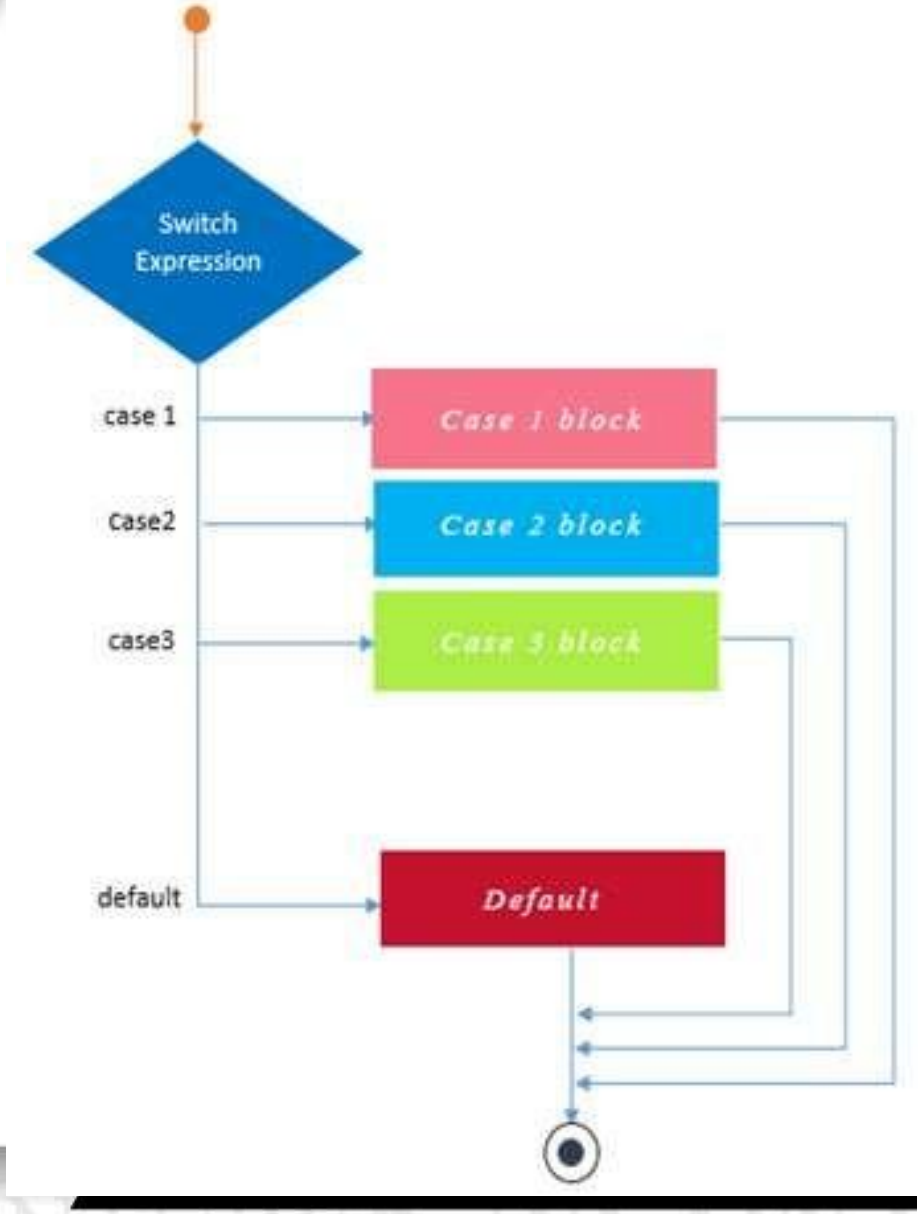
The Java switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement. The switch statement works with byte, short, int, long, enum types, String and some wrapper types like Byte, Short, Int, and Long. Since Java 7, you can use strings in the switch statement.

In other words, the switch statement tests the equality of a variable against multiple values.

1) There can be one or N number of case values for a switch expression.
2) The case value must be of switch expression type only. The case value must be literal or constant. It doesn't allow variables.
3) The case values must be unique. In case of duplicate value, it renders compile-time error.
4) The Java switch expression must be of byte, short, int, long (with its Wrapper type), enums and string.
5) Each case statement can have a break statement which is optional. When control reaches to the break statement, it jumps the control after the switch expression. If a break statement is not found, it executes the next case.
6) The case value can have a default label which is optional.

Check Program: Unit – 2 → SwitchDemo.java

Java Switch Statement



Check Program: Unit – 2 → SwitchDemo.java

Java Switch Statement


```
public class SwitchDemo{
        public static void main(String[] args)
        {
                int day = 5;
                String dayString;

                // switch statement with int data type
                switch (day) {
                case 1:
                        dayString = "Monday";
                        break;
                case 2:
                        dayString = "Tuesday";
                        break;
                case 3:
                        dayString = "Wednesday";
                        break;
                default:
                        dayString = "Invalid day";
                }
                System.out.println(dayString);
        }
}
```

Loops in Java

Loops can execute a block of code as long as a specified condition is reached.

Loops are handy because they save time, reduce errors, and they make code more readable.

Java Supports :-

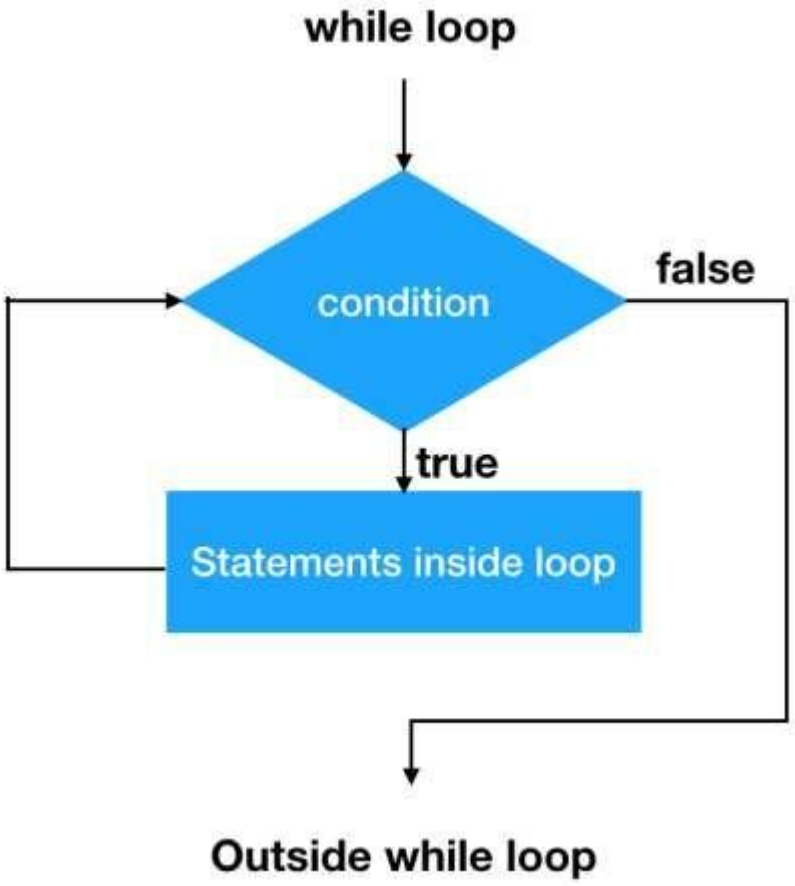
- While loop
- Do-while loop
- for loop

Java While Loop

The while loop loops through a block of code as long as a specified condition is true:

Syntax

```
while (condition) {
  // code block to be executed
}
```



Check Program: Unit – 2 → WhileDemo.java

Java While Loop


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

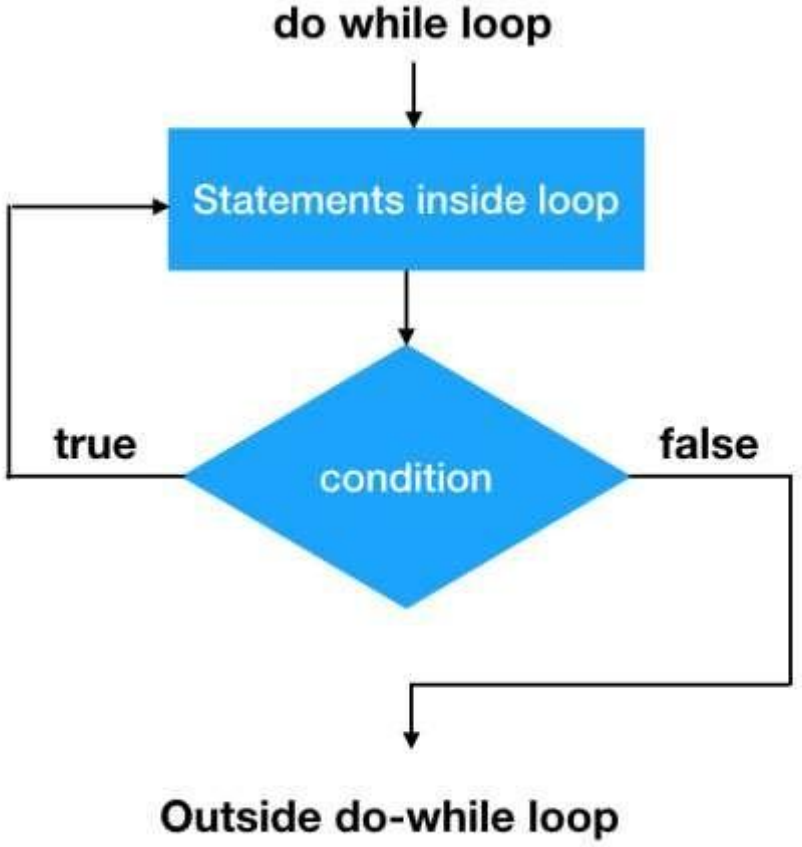
```
class whileLoopDemo {
   public static void main(String args[])
   {
      // initialization expression
      int i = 1;

      // test expression
      while (i < 6) {
         System.out.println("Hello World");

         // update expression
         i++;
      }
   }
}
```

Java do.. While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {
  // code block to be executed
}
while (condition);
```



Check Program: Unit – 2 → DoWhileDemo.java

Java do.. While Loop

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

```
public class DoWhileExample2 {
public static void main(String[] args) {
   do{
      System.out.println("infinitive do while
loop");
   }while(true);
}
}
```

Check Program: Unit – 2 → DoWhileDemo.java

Java for Loop

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

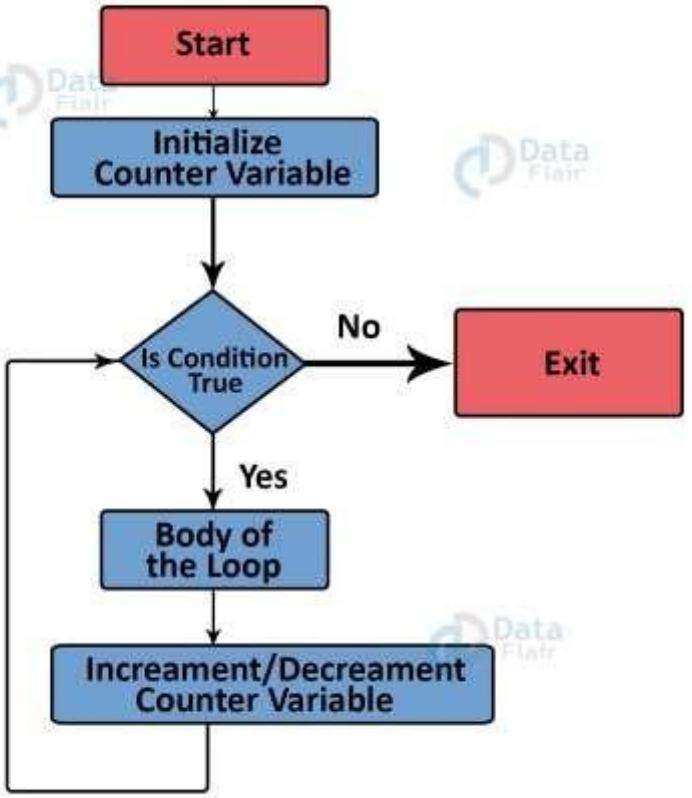
Syntax

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.



Check Program: Unit – 2 → ForDemo.java

Java for Loop

```
class Test {
   public static void main(String[] args) {

      // for loop
      for (int i = 1; i <= 10; ++i) {

         // if the value of i is 5 the loop terminates
         if (i == 5) {
            break;
         }
         System.out.println(i);
      }
   }
}
```

Check Program: Unit – 2 → ForDemo.java

Java Scanner, If, While


```
import java.util.Scanner;

class UserInputSum {
   public static void main(String[] args) {

      Double number, sum = 0.0;

      // create an object of Scanner
      Scanner input = new Scanner(System.in);

      while (true) {
         System.out.print("Enter a number: ");

         // takes double input from user
         number = input.nextDouble();

// if number is negative the loop terminates
         if (number < 0.0) {
            break;
         }

         sum += number;
      }
      System.out.println("Sum = " + sum);
   }
}
```

Check Program: Unit – 2 → ForDemo.java

Java Break statement

hen a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

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.

Check Program: Unit – 2 → BreakDemo.java

Java Break statement

```
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                    1
         break;                                 2
      }                                         3
      System.out.println(i);                    4
   }
}
}
```

Java continue statement

The continue statement is used in loop control structure when you need to jump/skip 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.

Cannot be used with Switch Statement

Check Program: Unit – 2 → ContinueDemo.java

Java continue statement


```
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);
   }
}
}
```

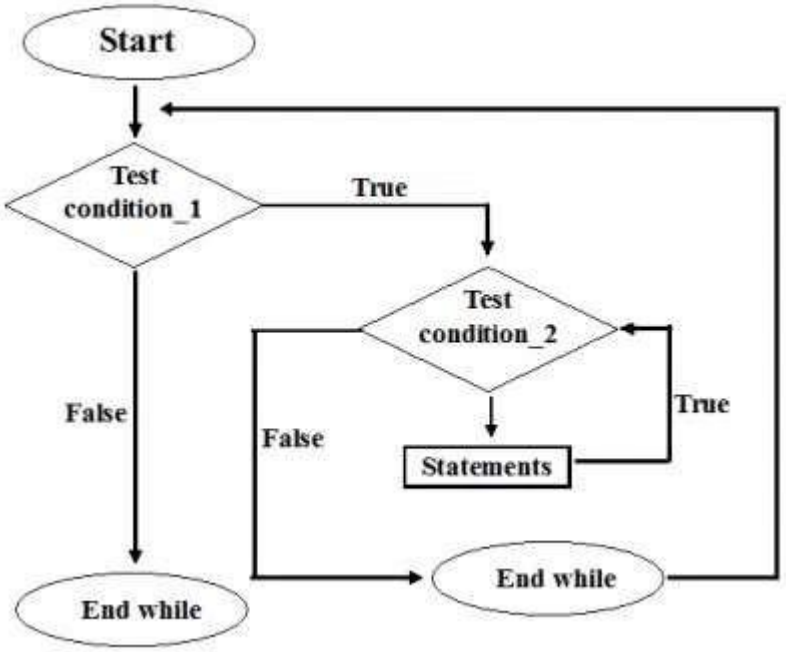
```
1
2
3
4
6
7
8
9
10
```

Break	Continue
The break statement is used to terminate the loop immediately.	The continue statement is used to skip the current iteration of the loop.
break keyword is used to indicate break statements in java programming.	continue keyword is used to indicate continue statement in java programming.
We can use a break with the switch statement.	We can not use a continue with the switch statement.
The break statement terminates the whole loop early.	The continue statement brings the next iteration early.
It stops the execution of the loop.	It does not stop the execution of the loop.

Java Nested Loop

If a loop exists inside the body of another loop, it's called a nested loop. Here's an example of the nested for loop.

```
int row,column;
// outer  loop
row = 1;
while(row <= 3) {
column=1;
 // inner loop
 while(column <=row) {
                System.out.println(row + " ");
        Column++;
 }
  System.out.println("\n");
  row++;
}
OUTPUT:-
1
2 2
3 3 3
```



Check Program: Unit – 2 → NestedForDemo.java

Java Nested Loop


```
class Main {
  public static void main(String[] args) {

    int rows = 5;

    for (int i = 1; i <= rows; ++i) {

            for (int j = 1; j <= i; ++j) {
                    System.out.print(j + " ");
            }
      System.out.println("");
    }
  }
}
```

Common mathematical methods

Min, Max, abs , log, sqrt method

Ex. Math.max(2.5, 3) returns 3.0

Math.min(2.5, 4.6) returns 2.5

Math.abs(-2) returns 2

Math.abs(-2.1) returns 2.1

The **java.lang.Math** class contains various methods for performing basic numeric operations such as the logarithm, cube root, and trigonometric functions etc.

Check Program : Unit – 2 → MathRandomDemo1.java

Common mathematical methods

Method	*Description*
`ceil(x)`	x is rounded up to its nearest integer. This integer is returned as a double value.
`floor(x)`	x is rounded down to its nearest integer. This integer is returned as a double value.
`rint(x)`	x is rounded up to its nearest integer. If x is equally close to two integers, the even one is returned as a double value.
`round(x)`	Returns (int)Math.floor(x + 0.5) if x is a float and returns (long)Math.floor(x + 0.5) if x is a double.

Rounding methods

Ex. Math.ceil(2.1) returns 3.0

Math.floor(2.8) returns 2.0

Math.rint(2.5) returns 2.0

Math.round(2.6f) returns 3

Check Program : Unit – 2 → MathRandomDemo3.java

Common mathematical methods

Random method : This method generates a random double value greater than or equal to 0.0 and less than 1.0 (0 <= Math.random() <1.0)

Ex.

(int)(Math.random() * 10) - - > Returns a random integer between 0 and 9.

a + Math.random() * b - - > Returns a random number between a and a + b,

excluding a + b.

Ex. 50 + (int)(Math.random() * 99) - - > Returns a random integer between 50 And 99.

Check Program : Unit – 2 → MathRandomDemo.java

Common mathematical methods

Trigonometric methods in Math class

Ex. Math.toDegrees(Math.PI / 2) returns **90.0**

Math.toRadians(30) returns **0.5236** (same as π/6)

Math.atan(**1.0**) returns **0.785398** (same as π/4)

Method	*Description*
`sin(radians)`	Returns the trigonometric sine of an angle in radians.
`cos(radians)`	Returns the trigonometric cosine of an angle in radians.
`tan(radians)`	Returns the trigonometric tangent of an angle in radians.
`toRadians(degree)`	Returns the angle in radians for the angle in degree.
`toDegree(radians)`	Returns the angle in degrees for the angle in radians.
`asin(a)`	Returns the angle in radians for the inverse of sine.
`acos(a)`	Returns the angle in radians for the inverse of cosine.
`atan(a)`	Returns the angle in radians for the inverse of tangent.

Check Program : Unit – 2 → MathRandomDemo2.java

Common mathematical methods


```
public class MathRandomDemo1
{
   public static void main(String[] args)
   {
      double x = 28;
      double y = 4;

      // return the maximum of two numbers
      System.out.println("Maximum number of x
                  and y is: " +Math.max(x, y));

      // return the square root of y
      System.out.println("Square root of y is: " + Math.sqrt(y));

      //returns 28 power of 4 i.e. 28*28*28*28
      System.out.println("Power of x and y is: " + Math.pow(x, y));

      // return the logarithm of given value
      System.out.println("Logarithm of x is: " + Math.log(x));
      System.out.println("Logarithm of y is: " + Math.log(y));

      // return the logarithm of given value when base is 10
      System.out.println("log10 of x is: " + Math.log10(x));
      System.out.println("log10 of y is: " + Math.log10(y));

      // return the log of x + 1
      System.out.println("log1p of x is: " +Math.log1p(x));

      // return a power of 2
      System.out.println("exp of a is: " +Math.exp(x));

      // return (a power of 2)-1
      System.out.println("expm1 of a is: " +Math.expm1(x));
   }
}
```

Check Program : Unit – 2 → MathRandomDemo1.java

Common mathematical methods


```
public class JavaMathExample2
{
   public static void main(String[] args)
   {
      double a = 30;

      // converting values to radian
      double b = Math.toRadians(a);

      // return the trigonometric sine of a
      System.out.println("Sine value of a is: " +Math.sin(a));

      // return the trigonometric cosine value of a
      System.out.println("Cosine value of a is: " +Math.cos(a));

      // return the trigonometric tangent value of a
      System.out.println("Tangent value of a is: " +Math.tan(a));

      // return the trigonometric arc sine of a
      System.out.println("Sine value of a is: " +Math.asin(a));

      // return the trigonometric arc cosine value of a
      System.out.println("Cosine value of a is: "
+Math.acos(a));

      // return the trigonometric arc tangent value of a
      System.out.println("Tangent value of a is: "
+Math.atan(a));

      // return the hyperbolic sine of a
      System.out.println("Sine value of a is: " +Math.sinh(a));

      // return the hyperbolic cosine value of a
      System.out.println("Cosine value of a is: "
+Math.cosh(a));

      // return the hyperbolic tangent value of a
      System.out.println("Tangent value of a is: "
+Math.tanh(a));
   }
}
```

Check Program : Unit – 2 → MathRandomDemo2.java

Common mathematical methods


```
public class MathRandomDemo3 {
 public static void main(String args[]) {
int i1 = 27;
  int i2 = -45;
System.out.println("Absolute value of i1: " + Math.abs(i1));

  System.out.println("Absolute value of i2: " + Math.abs(i2));

double d1 = 84.6;
  double d2 = 0.45;
  System.out.println("Round off for d1: " + Math.round(d1));

  System.out.println("Round off for d2: " + Math.round(d2));

System.out.println("Ceiling of '" + d1 + "' = " + Math.ceil(d1));
System.out.println("Ceiling of '" + d2 + "' = " + Math.ceil(d2));
 System.out.println("Floor of '" + d1 + "' = " +
Math.floor(d1));
 System.out.println("Floor of '" + d1 + "' = " +
Math.floor(d2));

System.out.println("exp(" + d2 + ") = " + Math.exp(d2));

  System.out.println("log(" + d2 + ") = " + Math.log(d2));

  System.out.println("pow(5, 3) = " + Math.pow(5.0, 3.0));

  System.out.println("sqrt(16) = " + Math.sqrt(16));
}
```

```
System.out.println("Minimum out of '" + i1 + "' and '" + i2 + "' = " + Math.min(i1, i2));

  System.out.println("Maximum out of '" + i1 + "' and '" + i2 + "' = " + Math.max(i1, i2));
```

Check Program : Unit – 2 → MathRandomDemo3.java

END OF UNIT - 2