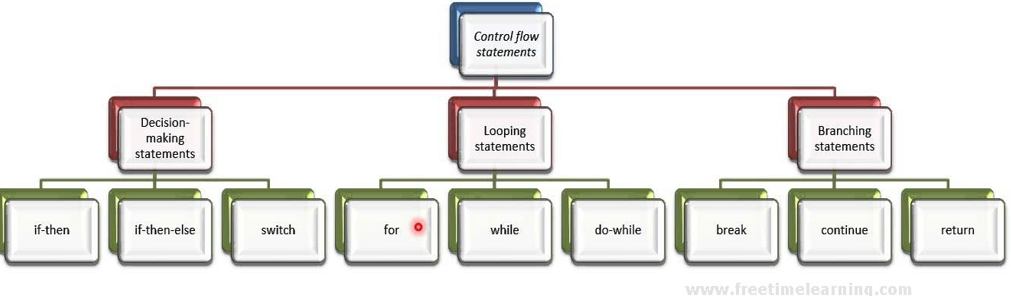
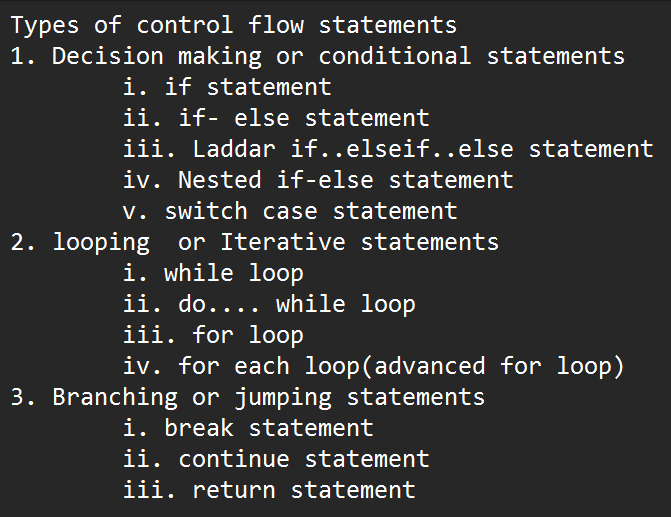
# **Java Notes**

1. Control Statements





1. Looping Statements

**When should we use looping statements?**

* When we want to repeat the action. We should use looping statements.

Types of Looping statements

* 1. While looping

**Syntax :**

Initialize loop

while (condition)

{

statements;

increment/decrement the loop

}

Sample Program:

class WhileDemo

{

public static void main(String arg[])

{

int x = 1; // initialization

while( x <= 5 ) // condition

{

System.out.println("x = "+ x);

x++; // iteration i.e incrementing loop

// x =x+1 🡪equal to 🡪 x++

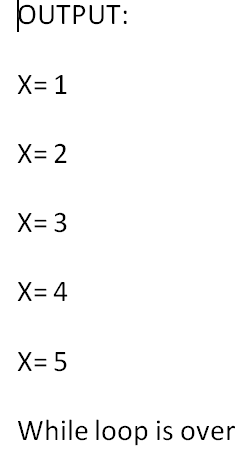
}

System.out.println(“While loop is over”);

}

}

**Output:**

****

* 1. do While looping.

**Syntax :**

Initialization statement

do {

// body of loop

Increment/decrement

} while (condition);

**Sample Program:**

**class DoWhileDemo**

**{**

**public static void main(String arg[])**

**{**

**int x = 1; // initialization**

**do**

**{**

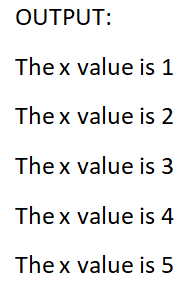
**System.out.println("The x value is " + x);**

**x++;**

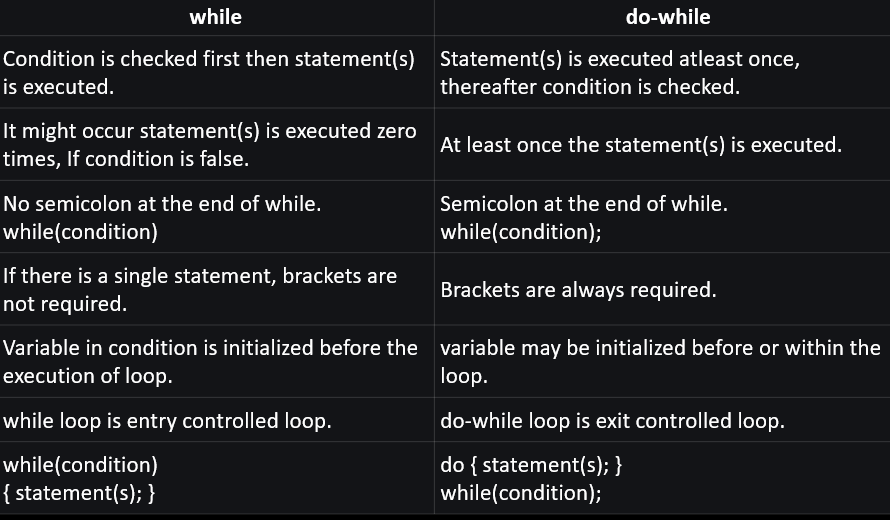
**} while( x <= 5 ); // condition**

**}**

**}**

****

**Difference between while and do..while loop**

****

* 1. For looping

**Syntax:**

for (initialization; condition; iteration🡪 increment/decrement)

{

    // body

}

**Sample Program:**

class ForExample

{

public static void main(String arg[])

{

for(int x = 1; x <= 5; x++)

{

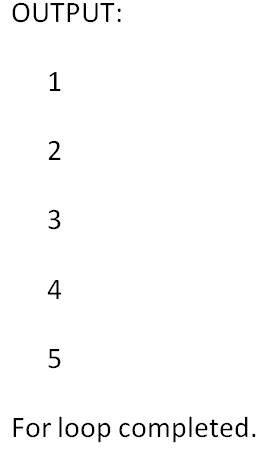
System.out.println(x);

}

System.out.println(“For loop completed.");

}

}



* 1. **Break Statement**
* If the condition is true i.e occurs, it will break the loop i.e come out of loop or break the current block of case in switch statement.

In java, break statement is majorly used for:

* Terminate a sequence in a switch- case statement
* To exit a loop intermediately.
* Used as another form of goto

**Sample Program:**

class BreakLoopDemo

{

public static void main(String args[])

{

// Initially loop is set to run from 0-9

for (int i = 1; i <=10; i++)

{

// terminate loop when i is 5.

if (i == 5)

break;

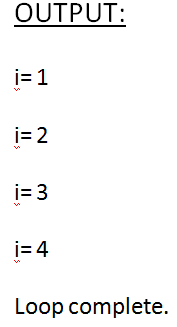
System.out.println("i= " + i);

}

System.out.println("Loop complete.");

}

}



* 1. Continue Statement
     + If the condition is true i.e occurs, it will skip current statements after this continue i.e continue to next loop.

Sample Program:

class PrintEvenNumbers  
{  
  public static void main(String arg[])  
  {  
   for(int i = 1; i < 10; i++)  
   {

      if(i == 5 )

continue;  
     System.out.println(i = “ + i);

}      
 }

}

**Output:**

i = 1

i = 2

i = 3

i = 4

i = 6

i = 7

i = 8

i = 9

* 1. Nested Looping

1. Functions or Methods
   1. Definition of Function
   2. Use or Advantages of functions
   3. Default functions or Functions without Arguments
   4. Parameterized Functions or Functions with Arguments
   5. Functions without Return
   6. Functions with Return
2. Branching Statements
   1. Break Statements
   2. Continue Statements
   3. Return Statements

* *Return statement is used to explicitly return from a method.*
* *It causes the program control to transfer back to the caller of the method.*

1. Array
   1. One Dimensional Array
   2. Two-Dimensional Array
2. String
   1. String Introduction
   2. String built-in methods
   3. String Buffer and String Builder class
3. Object Oriented Concepts
   1. Class and objects
4. Members of class
   1. Variables
   2. Methods
      1. Static Methods
      2. Non-Static Methods
   3. Constructors
5. Packaging
6. Access specifiers
   1. Public
   2. Default
   3. Private
   4. Protected
7. Encapsulation
8. Polymorphism
   1. Compile time polymorphism (Method overloading).
   2. Run time polymorphism (Method overriding).
9. Inheritance
   1. Single Inheritance
   2. Multi-level Inheritance
   3. Hierarchical Inheritance
   4. Multiple Inheritance
   5. Hybrid Inheritance
10. Abstraction
11. Interface
12. Exception Handling
    1. Definition of Exception
    2. Type of Exception
       1. Checked Exceptions or Compile Time Exception
       2. Unchecked Exceptions or Run Time Exception
    3. Exception Handling ways
       1. Try … catch
       2. Throws
    4. Use of finally block