Java Access Modifiers

Default Access Modifier

- Default access modifier means we do not explicitly declare an access modifier for a class, field, method, etc.
- A variable or method declared without any access control modifier is available to any other class in the same package. The fields in an interface are implicitly public static final and the methods in an interface are by default public.

Program N0. 9

Class Student

```
class Student
{
   int sid;    //default members
   String sname;

   void showStud()
   {
        System.out.println("\n\t Roll No :" + sid);
        System.out.println("\n\t Name :" + sname);
   }
}
public class DefaultAccessDemo_9
{
        public static void main(String[] args)
        {
                Student s1 = new Student();
                s1.sid = 101;
                s1.sname = "Prakash";
                s1.showStud();
        }
}
```

Public Access Modifier

- A class, method, constructor, interface etc declared public can be accessed from any other class.
- Therefore fields, methods, blocks declared inside a public class can be accessed from any class belonging to the Java Universe.

Program N0. 10

Class Employee

```
class Employee
{
    public int empid;
    public String ename;

    private void showEmp()
    {
        System.out.println("\n\t Empid :" + empid);
        System.out.println("\n\t Emp Name :" + ename);
    }
}

public class PublicModifierDemo
```

```
{
   public static void main(String[] args)
   {
        Employee e1 = new Employee();
        e1.empid = 101;
        e1.ename = "Sneha";
        e1.showEmp();
   }
}
```

Private Access Modifier

- Methods, Variables and Constructors that are declared private can only be accessed within the declared class itself
- Variables that are declared private can be accessed outside the class if public getter methods are present in the class.

Protected Access Modifier

- Variables, methods and constructors which are declared protected in a superclass can be accessed only by the subclasses in other package or any class within the package of the protected members' class.

Java Non Access Modifiers

Static Variables:

- The static key word is used to create variables that will exist independently of any instances created for the class.
- Only one copy of the static variable exists regardless of the number of instances of the class
    - Static variables are also known as class variables. Local variables cannot be declared static

Program N0. 11

Static Variable Demo

```
class Demo
{
   int x, y;
   static int count;

   void setxy()
   {
       X = 1;
       Y = 1;
       count++;
   }
}
public class StaticDemo1
{
   public static void main(String[] args)
   {
       System.out.println("\n\t No. of Objects = " +
       Demo.count);

       Demo d1 = new Demo();
       d1.setxy();

       Demo d2 = new Demo();
```

```
        d2.setxy();

        System.out.println("\n\t No. of Objects = " +
        Demo.count);
    }
}
```

Static Methods

- The static key word is used to create methods that will exist independently of any instances created for the class
- Static methods do not use any instance variables of any object of the class they are defined in. Static methods take all the data from parameters and compute something from those parameters, with no reference to variables
- Class variables and methods can be accessed using the class name followed by a dot and the name of the variable or method

Program N0. 12

Class Product (Static data members and static methods)

```
class Product
{
    int pid, price, qty;
    static double taxRate;

    static void setTax()
    {
        pid = 101;
        taxRate = 5.00;
    }
    static double getTax()
    {
        return taxRate;
    }
}
public class StaticDemo2
{
    public static void main(String[] args)
    {
        Product.setTax();
        System.out.println("Tax = " + Product.getTax());

    }
}
```

Final Modifier

Final Variables

- A final variable can be explicitly initialized only once.
- A reference variable declared final can never be reassigned to refer to a different object
- However the data within the object can be changed. So the state of the object can be changed but not the reference

Program N0. 13

Class Circle

```
class Circle
{
        double rad;
        final double PI=3.14;

        void setRadius(double r)
        {
                rad = r;
        }
        double getArea()
        {
                return PI*rad*rad;
        }

}
public class FinalVarDemo1_13
{
   public static void main(String[] args)
   {
        Circle cir1 = new Circle();
        cir1.setRadius(4.30);
        System.out.println("\n\t Area of cir1 = " + cir1.getArea());

   }
}
```

Program N0. 14

Final Object

```
public class FinalVarDemo_14
{
   public static void main(String[] args)
   {
        String str1 = "Welcome";

        System.out.println("\n\t str1 = " + str1);

        final String str2 = "Hello World";

        System.out.println("\n\t str2 = " + str2);

        str1 = "Welcome to Java";
        System.out.println("\n\t str1 = " + str1);

        str2 = "This is New String";
        System.out.println("\n\t str2 = " + str2);
   }
}
```

Final Methods

- A final method cannot be overridden by any subclasses. As mentioned previously the final modifier prevents a method from being modified in a subclass
- The main intention of making a method final would be that the content of the method should not be changed by any outsider

Final Classes

- The main purpose of using a class being declared as final is to prevent the class from being sub classed
- If a class is marked as final then no class can inherit any feature from the final class

Scanner Class

- Scanner class is defined in **java.util** package, it used to perform input operations
- The Scanner class breaks the input into tokens using a delimiter which is whitespace by default
- It provides many methods to read and parse various primitive values

Commonly used methods of Scanner class:

Method	Description
pubic String next()	Returns the next token from the scanner
public String nextLine()	Moves the scanner position to the next line and returns the value as a string.
public byte nextByte()	Scans the next token as a byte
public short nextShort()	Scans the next token as a short value
public int nextInt()	Scans the next token as an int value
public long nextLong()	Scans the next token as a long value
public float nextFloat()	Scans the next token as a float value
public double nextDouble()	Scans the next token as a double value

Program N0. 15

Input Age using Scanner Class

```
import java.util.Scanner;
public class ScannerDemo1_15
{
  public static void main(String[] args)
  {
        Scanner scan = new Scanner(System.in);

        int age;

        System.out.print("\n\t Enter your Age :");
        age = scan.nextInt();
```

```
        System.out.println("\n\t Your Age :" + age);
  }
}
```

Program N0. 16

Input average marks, name, character using scanner class

```
import java.util.Scanner;
public class ScannerDemo2_16
{
  public static void main(String[] args)
  {
        Scanner scan = new Scanner(System.in);

        float avg;
        char ch;
        String sname;

        System.out.print("\n\t Enter Average Marks :");
        avg = scan.nextFloat();

        System.out.print("\n\t Enter Any Character :");
        ch = scan.next().charAt(0);

        System.out.print("\n\t Enter Your Name :");
        sname = scan.next();

        System.out.println("\n\t Average Marks :" + avg);
        System.out.println("\n\t Character :" + ch);
        System.out.println("\n\t Name :" + sname);
  }
}
```

Assignment

Program N0. 17: Sum of two numbers
Program N0. 18: Input roll no, name marks of 3 subjects and calculate total marks, Average marks of a student
Program N0. 19: Input radius of a circle and calculate area and circumference

Command Line Arguments

- A Java application can accept any number of arguments from the command line. This allows the user to specify configuration information when the application is launched.
- The user enters command-line arguments when invoking the application and specifies them after the name of the class to be run.
- For example, suppose a Java application called Sort sorts lines in a file. To sort the data in a file named friends.txt, a user would enter: **java Sort friends.txt**
- When an application is launched, the runtime system passes the command-line arguments to the application's main method via an array of Strings

Program N0. 20

Command line arguments display length of arguments

```
public class CmdLIneArgsDemo1
{
   public static void main(String[] a)
   {
        System.out.println("\n\t No. of arguments :" + a.length);
   }
}
```

Program N0. 21

Command line arguments – pass two arguments and display the same

```
public class CmdLIneArgsDemo2
{
   public static void main(String[] a)
   {
        System.out.println("\n\t No. of arguments :" + a.length);

        System.out.println("\n\t 1st Argument :" + a[0]);
        System.out.println("\n\t 2nd Argument :" + a[1]);
   }
}
```

Program N0. 22

Command line arguments – display all arguments using loop

```
public class CmdLIneArgsDemo3
{
   public static void main(String[] a)
   {
        System.out.println("\n\t No. of arguments :" + a.length);

        for(int i=0; i<a.length; i++)
        {
           System.out.println("\n\t a["+i+"]:" + a[i]);
        }
   }
}
```

- If an application needs to support a numeric command-line argument, it must convert a String argument that represents a number, such as "34", to a numeric value.

Program N0. 23

Parsing Numeric Command-Line Arguments

```
public class CmdLIneArgsDemo4
{
   public static void main(String[] args)
   {
        int n1, n2;
        n1 = Integer.parseInt(args[0]);
        n2 = Integer.parseInt(args[1]);

        int sum = n1+n2;

        System.out.println("\n\t n1 = " + n1);
        System.out.println("\n\t n2 = " + n2);
        System.out.println("\n\t sum = " + sum);
   }
```

```
}
```

Decision Making Statements

- The program executes instructions sequentially. Sometimes, a program requires checking of certain conditions in program execution
- Java provides various conditional statements to check condition and execute statements according conditional criteria
- Followings are the different conditional statements used in Java
  - If Statement
  - If-Else Statement
  - Nested If-Else Statement
  - Switch Case

The if Statement

- An if statement consists of a Boolean expression followed by one or more statements
- If the Boolean expression evaluates to true then the block of code inside the if statement will be executed. If not the first set of code after the end of the if statement (after the closing curly brace) will be executed.
  The syntax of an if statement is:

```
if(Boolean_expression)
{
   //Statements will execute   if
     the Boolean expression is
     true
}
```

Program N0. 24

Input two numbers and display the max

```
import java.util.Scanner;
public class IfDemo1
{
	public static void main(String[] args)
	{
		Scanner scan = new Scanner(System.in);
		int a, b;

		System.out.print("\n\t Enter a :");
		a = scan.nextInt();

		System.out.print("\n\t Enter b :");
		b = scan.nextInt();

		if(a>b)
			System.out.println("\n\t a is Max");

		if(b>a)
			System.out.println("\n\t b is Max");
	}
}
```

Program N0. 25

Input Average marks and display the result pass if avg is greater than or equal to 35, otherwise result is fail

```
import java.util.Scanner;
public class IfDemo2
{
	public static void main(String[] args)
	{
		Scanner scan = new Scanner(System.in);
		float avg;

		System.out.print("\n\t Enter Average Marks :");
		avg = scan.nextFloat();

		if(avg>=35)
		{
			System.out.println("\n\t You Are Pass");
		}
		else
		{
			System.out.println("\n\t You Are Fail");
		}
	}
}
```

Program N0. 26

Input a number and display whether it is positive, negative or zero

```
import java.util.Scanner;
public class IfDemo3
{
	public static void main(String[] args)
	{
		Scanner scan = new Scanner(System.in);
		int no;

		System.out.print("\n\t Enter a no. :");
		no = scan.nextInt();

		if(no>0)
			System.out.println("\n\t number is Positive");
		else if(no==0)
			System.out.println("\n\t number is Zero");
		else
			System.out.println("\n\t number is Negative");
	}
}
```

Program N0. 27

Input Average marks and display the appropriate grades

```
import java.util.Scanner;
public class IFDemo4_27
{
	public static void main(String[] args)
	{
		Scanner sc = new Scanner(System.in);
```

```
		float avg;

		System.out.print("\n\t Enter average marks :");
		avg = sc.nextFloat();

		if(avg>=75)
			System.out.println("\n\t Distinction");
		else if(avg>=60)
			System.out.println("\n\t First Class");
		else if(avg>=45)
			System.out.println("\n\t Second Class");
		else if(avg>=35)
			System.out.println("\n\t Pass");
		else
			System.out.println("\n\t Fail");
	}
}
```

Program N0. 28

Input gender and age of an employee. An employee is eligible for insurance in following conditions

1. **Employee is male and age is greater than 30**
2. **Employee is Female and age is greater that 25**

```
import java.util.Scanner;
public class IFDemo5_28 {
	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);
		String gender;
		int age;

		System.out.print("\n\t Enter Gender :");
		gender = sc.next();

		System.out.print("\n\t Enter Age :");
		age = sc.nextInt();

		if(gender.toLowerCase().equals("male"))
		{
			if(age>=30)
				System.out.println("\n\t Employee is Eligible for
                                                    Insurance");
			else
				System.out.println("\n\t Not Eligible");
		}
		else if(gender.equalsIgnoreCase("female"))
		{
			if(age>=25)
				System.out.println("\n\t Employee is Eligible for
                                                Insurance");
			else
				System.out.println("\n\t Not Eligible");
		}
		else
		{
```

```
                        System.out.println("\n\t Wrong input");
                }
        }
}
```

Assignments

Program N0. 29	**Input price and qty, calculate bill amount with discount of 5.00% if price is greater than or equal to 50**
Program N0. 30	**Input a number and display whether it is Odd or Even**
Program N0. 31	**Input an alphabet and check whether it is vowel or consonant**

Logical Operators

Logical AND (&&)

- This operator is used to evaluate 2 or more conditions or expressions with relational operators simultaneously. If all expressions to the left and right of the logical expression are TRUE then the whole compound expression is TRUE

Program N0. 32

Input three numbers and display max

```
import java.util.Scanner;

public class IFDemo5_28
{
        public static void main(String[] args)
        {
                Scanner sc = new Scanner(System.in);

                int a, b, c;

                System.out.print("\n\t Enter a :");
                a = sc.nextInt();

                System.out.print("\n\t Enter b :");
                b = sc.nextInt();

                System.out.print("\n\t Enter c :");
                c = sc.nextInt();

                if(a>b && a>c)
                {
                    System.out.println("\n\t a is max");
                }
                else if(b>a && b>c)
                {
                    System.out.println("\n\t b is max");
                }
                else
                {
                    System.out.println("\n\t c is max");
```

```
                }
        }
}
```

Logical OR(||)

- This operator is used to evaluate or combine 2 or more expressions or conditions, and evaluates to TRUE if any one of all the expressions is true

Program N0. 33

Input any alphabet and display whether it is a vowel or a consonant

```
import java.util.Scanner;

public class IfDemo7
{
        public static void main(String[] args)
        {
                Scanner scan = new Scanner(System.in);
                char ch;
                System.out.print("\n\t Enter any alphabet :");
                ch = scan.next().charAt(0);

                if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u')
                                System.out.println("\n\t It is a Vowel");
                else
                                System.out.println("\n\t It is Consonant");
        }
}
```

Switch Statement

- The Switch Case Statement is used to make a decision from the number of choices
- The expression following the keyword switch is any C expression that evaluates an integer or a char value
- First, the expression following the keyword switch is evaluated. The value is then matched, one by one, against the constant values that follow the case statements. When a match is found, the program executes the statements following that case, and all subsequent case and default statements as well.
- If no match is found with any of the case statements, only the statements following the default are executed. A few examples will show how this control structure works.
- Expressions can also be used in cases provided they are constant expressions. Thus **case 3 + 7** is correct, however, **case a + b** is incorrect
- The **break** statement when used in a **switch** takes the control outside the **switch**. However, use of **continue** will not take the control to the beginning of **switch** as one is likely to believe

Program N0. 34

Switch Demo

```
public class SwitchDemo1 {
      public static void main(String[] args) {
            int no;
            no = 2;
            switch(no)
            {
            case 1:
```

```
                System.out.println("\n\t It is Number 1");
                break;
            case 2:
                System.out.println("\n\t It is Number 2");
                break;
            default:
                System.out.println("Invalid");
            }
        }
}
```

Program NO. 35

Addition..Multiplication.. Menu Program

```
import java.util.Scanner;
public class SwitchDemo2 {
      public static void main(String[] args) {
            Scanner scan = new Scanner(System.in);
            int num1, num2, ans,choice;
            System.out.print("\n\t Enter Num1 :");
            num1 = scan.nextInt();
            System.out.print("\n\t Enter Num2 :");
            num2 = scan.nextInt();
            System.out.println("\n\t\t1. Addition \n\t\t 2.
Subtraction
                  \n\t\t 3. Multiplication \n\t\t 4. Division");
            System.out.print("\n\t Enter a choice : ");
            choice = scan.nextInt();
            switch(choice)
            {
            case 1:
                  ans = num1+num2;
                  System.out.println("\n\t Addition = " + ans);
                  break;
            case 2:
                  ans = num1-num2;
                  System.out.println("\n\t Subtraction = " + ans);
                  break;
            case 3:
                  ans = num1*num2;
                  System.out.println("\n\t Multiplication = " +
                 ans);
                  break;
            case 4:
                  ans = num1/num2;
                  System.out.println("\n\t Division = " + ans);
                  break;
            default:
                  System.out.println("\n\t Wrong Input");
```

```
            }
      }
}
```

Program N0. 36

Pizza Order

```
import java.util.Scanner;
public class SwitchDemo3 {
      public static void main(String[] args) {
            Scanner scan = new Scanner(System.in);
            char pizzasize;
            System.out.println("\n\t Enter Pizza Size[s/m/l]:");
            pizzasize = scan.next().charAt(0);

            switch(pizzasize)
            {
            case 's':
                  System.out.println("\n\t Pizza size : Small");
                  break;
            case 'm':
                  System.out.println("\n\t Pizza size : Medium");
                  break;
            case 'l':
                  System.out.println("\n\t Pizza size : Large");
                  break;
            default:
                  System.out.println("\n\t Invalid Input");
            }
      }
}
```

Assignments

Program No. 37	If cost price and selling price of an item is input through the keyboard, write a program to determine whether the seller has made profit or incurred loss. Also determine how much profit he made or loss he incurred.
Program No. 38	Any year is input through the keyboard. Write a program to determine whether the year is a leap year or not. (Hint: Use the % (modulus)
Program No. 39	Write a program to check whether a triangle is valid or not, when the three angles of the triangle are entered through the keyboard. A triangle is valid if the sum of all the three angles is equal to 180 degrees
Program No. 40	A certain grade of steel is graded according to the following conditions: (i) Hardness must be greater than 50 (ii) Carbon content must be less than 0.7 (iii) Tensile strength must be greater than 5600 The grades are as follows: Grade is 10 if all three conditions are met

	Grade is 9 if conditions (i) and (ii) are met Grade is 8 if conditions (ii) and (iii) are met Grade is 7 if conditions (i) and (iii) are met Grade is 6 if only one condition is met Grade is 5 if none of the conditions are met Write a program, which will require the user to give values of hardness, carbon content and tensile strength of the steel under consideration and output the grade of the steel
Program No. 41	A library charges a fine for every book returned late. For first 5 days the fine is 50 paise, for 6-10 days fine is one rupee and above 10 days fine is 5 rupees. If you return the book after 30 days your membership will be cancelled. Write a program to accept the number of days the member is late to return the book and display the fine or the appropriate message.
Program No. 42	The policy followed by a company to process customer orders is given by the following rules: a. If a customer order is less than or equal to that in stock and has credit is OK, supply has requirement. b. If has credit is not OK do not supply. Send him intimation. c. If has credit is Ok but the item in stock is less than has order, supply what is in stock. Intimate to him data the balance will be shipped. Write a C program to implement the company policy.

Java Loops

- There may be a situation when we need to execute a block of code several number of times, and is often referred to as a loop.
- Java has very flexible three looping mechanisms. You can use one of the following three loops:
  - while Loop
  - do...while Loop
  - for Loop
  - As of Java 5, the enhanced for loop was introduced. This is mainly used for Arrays.

The while Loop

- A while loop is a control structure that allows you to repeat a task a certain number of times.
- When executing, if the boolean_expression result is true, then the actions inside the loop will be executed. This will continue as long as the expression result is true.
- Here, key point of the while loop is that the loop might not ever run. When the expression is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

Syntax:

```
while(Boolean_expression)
{
   //Statements
}
```

Program N0. 43

Print helloworld 5 times

```
public class WhileLoopDemo1 {
      public static void main(String[] args) {
            int i;

            i = 1;
            while(i<=5)
            {
                  System.out.println("\n\t Hello World");
```

```
                    i++;
                }
        }
}
```

Program N0. 44

Input 10 Number using While loop

```
import java.util.Scanner;
public class WhileDemo_44
{
        public static void main(String[] args)
        {
                Scanner scan = new Scanner(System.in);

                int no, i;
                i = 1;
                while(i<=10)
                {
                        System.out.print("\n\t Enter no :");
                        no = scan.nextInt();
                        i++;
                }
        }
}
```

Program N0. 45

Input 10 numbers and display count of positive, negative, zeros

```
import java.util.Scanner;
public class WhileDemo_45
{
        public static void main(String[] args)
        {
                Scanner scan = new Scanner(System.in);
                int no, i, pos=0, neg=0, zero=0;
                i = 1;

                while(i<=10)
                {

                        System.out.print("\n\t Enter no :");
                        no = scan.nextInt();
                        if(no>0)
                                pos++;
                        else if(no==0)
                                zero++;
                        else
                                neg++;
                        i++;
                }
```

```
                System.out.println("\n\t Total Positives :" + pos);
                System.out.println("\n\t Total Negatives :" + neg);
                System.out.println("\n\t Total Zeros :" + zero);
        }
}
```

do...while Loop

- A do...while loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time
- Notice that the Boolean expression appears at the end of the loop, so the statements in the loop execute once before the Boolean is tested.
- If the Boolean expression is true, the flow of control jumps back up to do, and the statements in the loop execute again. This process repeats until the Boolean expression is false.
-

Syntax:

```
Do
{
   //Statements
} while(Boolean_expression);
```

Program N0. 46

Do..while demo

```
public class DoWhileLoopDemo1
{
        public static void main(String[] args) {
                int i = 10;

                do
                {
                        System.out.println("Value of i = " + i);
                        i++;
                }while(i<=5);
        }
}
```


Program N0. 47

Input Record of Student until user enters “stop”

```
import java.util.Scanner;

public class WhileDemo_47
{
        public static void main(String[] args)
        {
                Scanner scan = new Scanner(System.in);
                int roll;
                 choice;

                do
                {
                        System.out.print("\n\n\t Enter Roll No. :");
                        roll = scan.nextInt();

                        System.out.print("\n\n\t Enter Name :");
```

```
                    name = scan.next();


                    System.out.print("\n\n\t Next Record? [ok=Next |stop   =
                    Cancel] :" );
                    choice = scan.next();


            }while(!choice.equalsIgnoreCase("stop"));
      }
}
```

For loop

```
Syntax:
for(initialization; Boolean_expression; update)
{
    //Statements
}
```

- A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
- A for loop is useful when you know how many times a task is to be repeated.
- The initialization step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
- Next, the Boolean expression is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement past the for loop.
- After the body of the for loop executes, the flow of control jumps back up to the update statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the Boolean expression.
- The Boolean expression is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then update step, then Boolean expression). After the Boolean expression is false, the for loop terminates

Program N0. 48

For Loop Demo

```
public class ForDemo_48
{
      public static void main(String[] args)
      {
            for(int i=0; i<5; i++)
            {
                  System.out.println("i = " + i);
            }
      }
}
```

Enhanced for loop in Java

```
Syntax:
for(declaration : expression)
{
    //Statements
}
```

- As of Java 5, the enhanced for loop was introduced. This is mainly used for Arrays.
- **Declaration:** The newly declared block variable, which is of a type compatible with the elements of the array you are accessing. The variable will be available within the for block and its value would be the same as the current array element.
- **Expression:** This evaluates to the array you need to loop through. The expression can be an array variable or method call that returns an array.

Program N0. 49

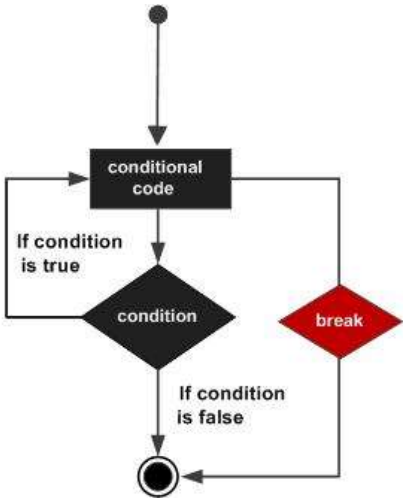
Enhanced For loop Demo

```
public class ForDemo_49{
	public static void main(String[] args)
	{
		System.out.println("\n\t Command line arguments are :");

		for(String item : args)
		{
			System.out.println(item);
		}
	}
}
```

The break Keyword

- The break keyword is used to stop the entire loop. The break keyword must be used inside any loop or a switch statement.

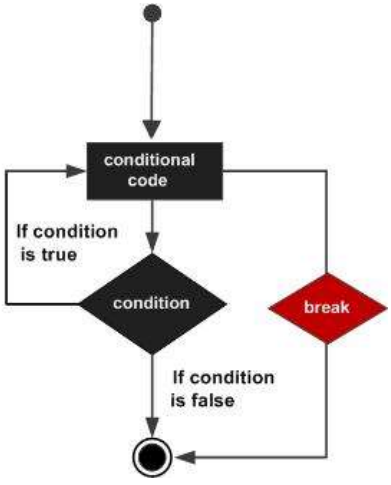


Program N0. 50

Break Statement

The continue Keyword

- The continue keyword can be used in any of the loop control structures. It causes the loop to immediately jump to the next iteration of the loop



Program NO. 51

Continue Statement

```
/* Java49. WAP to display the numbers upto 100 by using continue
keyword to continue or break */

import java.util.Scanner;
class Java49
{
	public static void main(String []args)
	{
		Scanner scan = new Scanner(System.in);

		int cnt,i;
		String result;

		cnt=0;
		for(i=1;i<=100;i++)
		{
			System.out.print("  " + i);
			cnt++;
			if(cnt==10)
			{
				cnt=0;
				System.out.println("\n\t Do you to Continue
					[yes/no] :");
				result = scan.next();

				if(result.equals("yes"))
				{
					continue;
				}
				else
				{
					break;
				}
			}
		}
	}
}
```