Core Java by Vaibhav Sir

Contents

Page No.

Core Java - Day: 01

Java installation procedure:

Step1: Command to check java version: java -version

step2: check operating system: right click on This PC--> properties--> system type

Step3: how to download java

search download java 1.8-->click on oracle.com-->Java SE Development Kit 8u281--> click on 32/64 related file to download java

Step4: install downloaded java file

Step5: set java path: 5A: Copy java path

open C drive-->program files-->java-->jdk/jre(jdk preferred)-->bin folder-->copy bin folder path

Step5A: Set java path

right click on This Pc/my computer-->properties-->advanced system setting-->environment variable-->

user variable--> check for "path" variable

--Case1: already path variable exists -->click on path variable-->edit-->new-->ctrl+ V(Paste) --> ok--ok

--Case2: no path variable--> click on new --> enter variable name i.e. -"path"--> variable value -ctrl+ V(Paste)--> ok-->ok

---

Day: 02

Diff versions eclipse IDE:

versions: oxygen neon marsh

photon(latest) --> java version 11 & above

eclipse installation procedure:

Step1:

-->search download eclipse oxygen-->click on https://www.eclipse.org/-->MORE DOWNLOADS-->Eclipse 2020-12 (4.14)-->

Eclipse IDE for Java and DSL Developers-->Windows x86_64--> eclipse-dsl-2020-12-R-win32-x86_64.zip

Step2:

--open downloads folder --> unzip eclipse file ---> open eclipse folder--> double click on eclipse application file (blue colour icon)-->

it should ask for workspace path--> keep as it is(default path)-->select checkbox-->launch--> welcome page

---

Day 03

Main Method

```
package sample1;
public class demo1
{
        //class body
        public static void main(String [] args) //main method declaration
        {
                System.out.println("Hi.."); //printing statement
                System.out.println("hello..");
                System.out.println("Hi..");
        }
}
```

```
package sample1;

public class demo2
{

        public static void main(String[] args) {

                System.out.println("Hi");
        }

}
```

Java 1st program:

1. Create java project
2. create java package
3. create java class
4. main method
5. printing statemnt
6. save program (Control + s)
7. run program (click green btn)
8. check output--> Console tab

Shortcuts in eclipse:

1. main method: type "main"+control+space
2. Printing statement: type "syso"+control+space

String, System--> Caps

Step1: create java project--> file--> new--> java project--> enter project name--> finish

Step2: create java package--> right click on project name/Src folder-->new--> package--> enter package name--> finish

Step3: create java class-->right click on package name-->new-->class-->enter class name--> finish

1 java project--> multiple packages-->

1 packages--> multiple classes

1 class--> multiple method--> main method

1 main method --> multiple printing statements --> to print messages

---

```
package Variables;
public class demo1
{
	public static void main(String[] args) {
		// String --> more than 1 character-->	abc, rahul, abc@123, #$%,
		// int		--> numeric + non-decimal --> 100, 5, 15, 10000
		// float		--> numeric + decimal	->56.5, 2.6,	10000.5
		// char		--> single character	--> A, c,d,b
	}
}
```

- **Variables:**

Variables are nothing but piece of memory use to store information.

one variable can store 1 information at a time

Variables also used in information reusability.

To utilize variables in java programing language we need to follow below steps:

1. Variable declaration (Allocating/Reserving memory)

2. Variable Initialization (Assigning or Inserting value)

3. Usage

Note:- According to all programming language dealing with information directly is not a good practice and to overcome this variable are introduced.

```
package Variables;
public class demo3 {
public static void main(String[] args)
{
        //step1: variable declaration
        String sname;
        int srollNum;
        char sgrade;
        float sper;
        //step2: variable initialization
        sname = "swapnil";
        srollNum = 10;
        sgrade ='A';
        sper= 65.5f;
        //step3: usage
        System.out.println("Student name: "+sname);
        System.out.println("Student roll num: "    +  srollNum);
        System.out.println("Student grade: "+sgrade);
        System.out.println("Student percentage: "+sper +"%");
}
}
```

• Data Types:

- **Data type are used to represent type of data or information which we are going to use in our java program.**
- **In java programming it is mandatory to declare datatype before declaration of variable.**

In java datatypes are classified into two types:

1. Primitive datatype.
2. Non-primitive datatype.

1.Primitive datatype:

There are 8 types of primitive datatypes.

all the primitive datatypes are keywords.

*** Memory size of primitive datatype are fix.**

The types of primitive datatype are:

Note: -Keyword starts with lower case
Primitive datatype starts with lower case

syntax: datatype variablename;

I (Numeric + Non-decimal): - Ex: 80,85, 10, etc.

	Data Type	Size
1.	byte	1 byte
2.	short	2 bytes
3.	int	4 bytes
4.	long	8 bytes

II (Numeric + decimal): - Ex: 22.5,22.8,6.4....

5.	float	4 byte
6.	double	8 byte

III Character: - Ex: A, B, X, Z.

7.	char	2 byte

IV Conditional: - Ex: true, false.

8.	boolean	1 bit

---

2. Non-primitive datatype:

There are 2 types of non-primitive datatypes.

all the Non primitive datatypes are identifiers.

*** Memory size of non-primitive datatype is not defined or not fix.**

Note: Identifier starts with capital letter.
Non-primitive datatype starts with capital letter.
e.g. String, className

- **Methods**

- **A method is a block of code which only runs when it is called.**
- **You can pass data, known as parameters, into a method.**
- **Methods are used to perform certain actions, and they are also known as functions.**

Why use methods? To reuse code: define the code once, and use it many times.

1. main method

- **In any Java program, the main () method is the starting point from where compiler starts program execution.**
- **So, the compiler needs to call the main () method.**
- **without main method we can't run any java program.**

2. Regular method

1. static regular method

1. static method call from same class --> methodname();

2. static method call from diffrent/another class -->className.methodname();

2. non- static regular method

3. non-static method call from same class --> create object of same class

4. non-static method call from diffrent/another class --> create object of diff class

Note: At the time of program execution main method is going to get executed automatically,

- **whereas regular methods are not going to get executed automatically.**
- **At the time of program execution priority is scheduled for main method only.**
- **To call a regular method we need to make call method call from main method, until unless if the method call is not made regular method will not get executed.**
- **Regular methods can be called multiple times.**

When You will use Static / Non-Static Method?

Static Method: If You want to maintain Constant Information of object throughout the classes then static method is used.

Eg: CEO for every employee is same i.e., CEO change then it will change for every employee.

Non-Static Method: For each object if you want different information then we will use non static method.

Eg: If Employee Changes Then Employee ID also Changes

```
//1. static regular method call from same class      --> methodname();
package Methods;
public class sample1 {
public static void main(String[] args)
{
        m1();     //methodname();        method call
        m2();
        m2();   //method reuse
}
// static-->regular method
public static void m1()     //regular method declaration
{
        System.out.println("running static regular method: m1");
}
//static-->regular method
public static void m2()     //regular method declaration
{
        System.out.println("running static regular method: m2");
}
}
```

```
//2. static method call from diffrent/another class   -->className.methodname();
package Methods;
public class sample2
{
public static void main(String[] args) {
        sample3.m3();    //classname.methodname();
        sample3.m4();
        sample3.m4();     //method reuse

}
}
```

Here Sample 3 is Another Class

```
package Methods;
public class sample3
{
        //static -->regular method
        public static void m3()
        {
                System.out.println("running static regular method m3 from diff class");
        }
        //static -->regular method
        public static void m4()
        {
                System.out.println("running static regular method m4 from diff class");
        }

}
```

---

1. A) static Regular method call from same class

static method call from same class

Sample1.java

```
1  package Methods;
2          // Example of Using Regular Method (Same Class)
3  public class Sample1
4  {                                                    // This is Main Method//
5      public static void main(String[] args) // Main Method Declaration
6      {
7
8          System.out.println("Main Method Started");
9
10         regular();          // Regular--> MethodName();     Method Call
11         regular2();
12
13         System.out.println("Main Method Ended");
14
15                                              // This is Regular Method//
16     }
17         public static void regular()        // Regular Method Declaration
18         {
19             System.out.println("Running Static Regular Method");
20         }
21
22         public static void regular2()
23         {
24             System.out.println("Running Another Regular Static Method");
25         }
26 }
27
```

1. B) Static Regular Method call from Different class

Class : 1st

Sample2.java | Sample3.java

```java
package Methods;

public class Sample2
{ public static void main(String[] args)


    {
        Sample3.AnotherSheet();
        Sample3.DifferentSheet();

    }

}

```

Class: 2nd

Sample2.java | Sample3.java

```java
package Methods;

public class Sample3
{
    public static void DifferentSheet()
    {
        System.out.println("Static Method Call For Different Class");
    }

    public static void AnotherSheet()
    {
        System.out.println("Static Method Call Another Example");
    }
}

```

```
//3. non-static method call from same class    --> create object of same class
package Methods;
public class sample4
{
public static void main(String[] args)
{
        //classname objectname=new classname();  --> object creation
        sample4 s4=new sample4();     // object creation
        s4.m5();                    //objectname.methodname();
        s4.m6();
        s4.m6();      //reuse

}
//non-static -->regular method
public void m5()
{
        System.out.println("running non-static regular method m5 from same class");
}
//non-static -->regular method
public void m6()
{
        System.out.println("running non-static regular method m6 from same class");
}

}
```

Non-static method Call from same class

```java
package Methods;
public class Nonstatic
{public static void main(String[] args)
    {
        // classname objectname = new classname(); // Object Creation

    Nonstatic o1 = new Nonstatic(); // object creation
                                    // objectname.methodname();
    o1.m5();
    o1.m6();
                                // Nonstatic = datatype
                                // o1 = objectname
    }                           // new = keyword to create blank object
                                // classname() --> Constructor
                                // Copy all members of class into object
// Non Static Regular Method
public void m5()
{
    System.out.println("Running Non Static Regular MEthod m5 from same class");
}

public void m6()
{
    System.out.println("Running Non Static Regular MEthod m6 from Same Class");
}

}
```

//4. Non-static method call from different/another class --> create object of diff class

Class: 1st

```
package Methods;
public class sample5
{
public static void main(String[] args)
{
        //classname objectname =new classname();
        sample6 s6=new sample6();   //object creation of diff class
        s6.m7();
        s6.m8();
        //sample6--> datatype
        //s6---> objectName
        //new--> keyword--> to create blank object
        //classname() --> constructor --> to copy all the members of class into object
}
}
```

Class: 2nd

```
package Methods;
public class sample6 {
        //non-static -->regular method
        public void m7()
        {
                System.out.println("running non-static regular method m7 from diff class");
        }
        //non-static -->regular method
        public void m8()
        {
                System.out.println("running non-static regular method m8 from diff class");
        }
}
```

Non-static method call from different/another class

Class: 1st

```
Sample2.java   Sample3.java   *Nonstatic.java   NonstaticAnother.java

1  package Methods;
2
3  public class NonstaticAnother
4  {
5      public static void main(String[] args)
6
7      {
8          // Classname objectname = new Classname();
9
10         NonstaticAnotherClass NAC = new NonstaticAnotherClass();
11         NAC.n1();
12         NAC.n2();
13
14     }
15
16 }
17
```

Class: 2nd

```
Sample2.java   Sample3.java   *Nonstatic.java   NonstaticAnother.java   NonstaticAnotherClass.java

1  package Methods;
2
3  public class NonstaticAnotherClass
4  {
5
6      public void n1()
7      {
8          System.out.println("Printing Non Static Regular Method n1 from Another Class");
9      }
10
11     public void n2()
12     {
13         System.out.println("Printing Non Static Regular Method n2 from Different Class");
14     }
15
16
17 }
18
```

Example Regarding

Static Regular Method Same Class

Non-Static Method from Same Class

Static Regular Method Same Class

Non-Static Regular Method Same Class

Class 1st

AllMethods.java | AllMethodsAnotherClass.java

```java
package Methods;

public class AllMethods
{
    public static void main(String[] args)
    {
        m1();                                   // Static Regular Method Same Class
        AllMethods NSMSC = new AllMethods();    // Object Creation
        NSMSC.m2();                             // Non Static Method From Same Class

        AllMethodsAnotherClass.Another1();      // Static Regular MEthod From Another CLass
        AllMethodsAnotherClass NSMAC = new AllMethodsAnotherClass(); // Object Creation
        NSMAC.Another2();                       // Non Static Method From Another Class
    }


    public static void m1()                     // Static Regular Method Same Class
    {
        System.out.println("1. Printing Static Regular Method m1 From Same Class");
    }

    public void m2()                            // Non Static Regular Method Same Class
    {
        System.out.println("2. Printing Non Static Regular Method m2 From Same Class");
    }
}
```

Class 2nd

AllMethods.java | AllMethodsAnotherClass.java

```java
package Methods;

public class AllMethodsAnotherClass
{
    public static void Another1()           // Static Regular Method From Another Class
        {
            System.out.println("3. Printing Static Regular Method Another1 From Another Class");
        }

    public void Another2()                  // Non Static Regular Method From Another Class
        {
            System.out.println("4. Printing Non Static Regular Method Another2 From Another Class");
        }
}
```

(Above methods are Method Without Parameter/

Method With Zero Parameter)

- **Method without Parameter**

Class 1st

```
package Methods;
public class sample7
{
	public static void main(String[] args) {
		m1();
		sample7 s7=new sample7();
		s7.m2();
		sample8.m3();
		sample8 s8=new sample8();
		s8.m4();
	}
	//static regular method -->without parameter
	public static void m1()
	{
		System.out.println("running static method m1 from same class");
	}
	//non-static regular method -->without parameter
	public  void m2()
	{
		System.out.println("running non-static method m2 from same class");
	}
}
```

Class 2nd

```
//method without/zero parameter

//static regular method -->without parameter

package Methods;

public class sample8 {

	public static void m3()

	{

		System.out.println("running static method m3 from diff class");

	}

	//non-static regular method -->without parameter

	public void m4()

	{

		System.out.println("running non-static method m4 from diff class");

	}

}
```

```
//example6:  Method with Parameter--> int parameter

package Methods;
public class sample9
{
        public static void main (String [] args)
        {
                addition(100, 200);
                addition(50, 25);
                sample9 s9=new sample9();
                s9.mul(10, 25);
        }
        //static regular--> method with parameter (2 int parameter) (int,int)
        public static void addition(int a, int b)
        {
                int sum =a+b;
                System.out.println(sum);
        }
        //non-static regular--> method with parameter(2 int parameter)(int,int)
        public void mul(int a, int b)
        {
                int mulValue =a*b;
                System.out.println(mulValue);
        }
}
```

Method With Parameter (int a, int b)

MethodWithParameter.java

```
1  package Methods;
2                                          // Method With Parameter
3  public class MethodWithParameter
4  {
5      public static void main(String[] args)
6      {
7          addition(2,3,4);
8          MethodWithParameter m1 = new MethodWithParameter();
9          m1.multiplication(4,5,6);
10
11     }
12
13     //1. Static Regular Method With Parameter from Same Class (2 int Parameter) (int,int)
14     public static void addition(int a, int b, int c)
15     {
16         int sum = a + b + c;
17         System.out.println(sum);
18     }
19
20     //2. Non static Regular Method With Parameter from Same Class (2 int parameter) (int,int)
21     public void multiplication(int a, int b, int c)
22     {
23         int mulvalue = a*b*c;
24         System.out.println(mulvalue);       // Dont use Double inverted Comma at mulvalue
25     }
26
27 }
```

```
//example6:  method with parameter--> String parameter

package Methods;

public class sample10

{

        public static void main(String[] args)

        {

                studentName("swapnil");

                studentName("abc");

        }

        public static void studentName(String sname)

        {

                System.out.println(sname);

        }

}
```

```
//example7:  method with parameter--> all types of parameter

package Methods;
public class sample11 {
public static void main(String[] args) {


        studentInfo("swati",200,'A',65.5f);

        System.out.println("-----------");

        studentInfo("amol",250,'B',68.5f);

}

public static void studentInfo(String sname,int sRollNum,char sgrade, float sper)

{

        System.out.println(sname);
        System.out.println(sRollNum);
        System.out.println(sgrade);
        System.out.println(sper);
}

}
```

Method With Parameter (String ABC)

*MethodWithStringParameter.java

```java
package Methods;
                                // Method With Parameter ( String name)
public class MethodWithStringParameter
{
    public static void main(String[] args)
    {
    studentname("Mr.Vaibhav Yendole");
    }
                                // Public Static Method With Parameter (String abc)
    public static void studentname(String sname)
    {
        System.out.println(sname);
    }

}
```

Method With Parameter – All Types of Parameters

MethodWithAllParameter.java

```java

public class MethodWithAllParameter
{
public static void main(String[] args)
    {
        StudentInfo("Anand", 007, 'A',81.83f);


    }

public static void StudentInfo(String Sname, int RollNo, char SGrade, float SPer)
    {
        System.out.println(Sname);
        System.out.println(RollNo);
        System.out.println(SGrade);
        System.out.println(SPer);
    }
}
```

*Sample100.java

```java
package Methods;

public class Sample100
{

public static void main(String[] args)
    {
        BikeInfo("Apache200","MH-12 CD 0001", 210, 32.5f, 'A' );
    }

public static void BikeInfo (String Bname, String NPlate, int CC,
                                float Avg , char Brake)
    {
        System.out.println(Bname);
        System.out.println(NPlate);
        System.out.println(CC);
        System.out.println(Avg);
        System.out.println(Brake);
    }

```

- **Loops**

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true.

There are 4 types of loops in Java.

1. for loop

2. while loop

3. do while

4. for each--array/collection--> selenium

```
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.

1.for loop

```
package Loops;                                  (Ascending Order)
public class example1_forLoop
 {
public static void main(String[] args)
{
        // 1(start position)    6<=5(End Position)   6 (No of Steps)
        for(int i=1; i<=5; i++)         // i++ For Increament i with one value
        {                               // i+2 for Increament I with Two Values
                System.out.println(i);    // 1 2 3 4 5
        }
}
}
```

```
package Loops;
public class example3_forLoop_print_Odd_Numbers_from_1_to_99
{
public static void main(String[] args)
 {
        for(int i=1; i<=99; i=i+2)
        {
                System.out.println(i);                  //1  3
        }
}
}
```

```
package Loops;                                  (Descending/Reverse Order)
public class example5_forLoop_print_numbers_from_5_to_1
{
public static void main(String[] args)
 {
                        // 5(Start)    0>=1(End)    0(Step)
        for(int i=5; i>=1; i--)
        {
                System.out.println(i);                  //5 4 3 2 1
        }
}
}
```

```
package Loops;
public class example6_forLoop_print_odd_numbers_from_99_to_1
{
public static void main(String[] args)
 {
        for(int i=99; i>=1; i=i-2)
        {
                System.out.println(i);          //5 4 3 2 1
        }
}
}
```

---

package Loops; (For Tables/ Multiples)

```
public class example8_forLoop
{

        public static void main(String[] args)
        {

                for(int i=1; i<=10; i++)
                {
                        System.out.println(i*2);
                }

        }

}
```

---

package Loops; (If We want to Print String Statement No of Times)

```
public class example10_forLoop_print_Msg_MultipleTimes
{
public static void main(String[] args)
{

        for(int i=1; i<=10; i++)
        {
                System.out.println("Hi");
        }

}
}
```

---

*ForAscendingOrder.java

```
1  package loops;
2
3  public class ForAscendingOrder
4  {
5      public static void main(String[] args)
6      {
7          for (int i=1; i<=10; i++)        // i++ or i+1 is Increament by 1
8                                           // i+2 --> Increament by 2
9          {
10             System.out.println(i);       // i--  = Decreament by 1
11
12         }                                // i-2  = Decreament by 2
13
14     }
15 }
16
```

```
<term
1
2
3
4
5
6
7
8
9
10
```

ForAscendingOrd... | Example2OddNo.java | Example4EvenNo.j... | Example5ReverseO... | Exampl

```java
package loops;
                    // For Printing Odd/ Even No in Ascending Order//
public class Example2OddNo
{
    public static void main(String[] args)
    {
        for (int i=1; i<=20; i=i+2) //( Start; End; Increament Steps)
        {
            System.out.println(i);
        }

    }

}
```

For Increment By 2 Use (I=i+2)

For Decrement by 1 Use (i--)

ForAscendingOrd... | Example2OddNo.java | Example4EvenNo.j... | *Example5Reverse... | Exa

```java
package loops;
                        // For Descending Order or Reverse
public class Example5ReverseOrder
{
    public static void main(String[] args)
    {
        for (int i=22; i>=2; i=i-2) // (Start; End; Decrement Step)

        {
            System.out.println(i);
        }
    }
}
// Warning : For Increment by 1 use (i++) Never Use (i=i++)
// Warning : For Decrement by 1 use (i--) Never Use (i=i--)
//           For Increment by 2 Use (i=i+2)
//           For Decrement by 2 Use (i=i-2)
```

Must Read the Above Warning

ForAscendingOrd... | Example2OddNo.java | Example4EvenNo.j... | Example5ReverseO... | Example6MultipleT...

```java
package loops;

public class Example6MultipleTimes
{
public static void main(String[] args)
{
    for (int i=1; i<=10; i++) // Warning For Increament by 1 Never Use (i=i++)
    {                                                    //Use Only  (i++)
        System.out.println("Use This Method Printing Multiple Times");
    }

}
}
```

2. while loop

```
package Loops;
public class example11_whileLoop
{
        public static void main(String[] args)
        {
                int i=1;                        //start condition
        while(i<=5)     //end condition         //6<=5
        {
                System.out.println(i);          //5
                i++;   //6          //increment or decrement
        }
        }
}
```

```
package Loops;
public class example13_whileLoop_print_even_num_from_2_to_100
{
public static void main(String[] args)
        {
        int i=2;          //start condition

        while(i<=100)     //end condtion
        {
                System.out.println(i);     //5
                i=i+2;            //6          //increment or decrement
        }
}
}
```

```
package Loops;
public class example14_whileLoop_print_odd_num_from_99_to_1
{
public static void main(String[] args)
{
        int i=99;         //start condition
        while(i>=1)     //end condtion
        {
                System.out.println(i);      //5
                i=i-2;   //6          //increment or decrement
        }
}
}
```

3. Do While Loop

```
package Loops;
public class example16_DoWhile
{
public static void main(String[] args)
    {
    int i=10;          //start condition
    do
    {
            System.out.println(i);          // 10
            i++;     //11                   // increment or decrement
    }
    while (i<=5);                           //end condition

    }
}
```

Example on (for loop)

```
package loops;

public class ForLoopExample
{
public static void main(String[] args)
{
    for (int i=1; i<=10; i++)
    {
        System.out.println("Table of 2 :"+i*2);
    }
}
}
```

```
<terminated> ForLoopExan
Table of 2 :2
Table of 2 :4
Table of 2 :6
Table of 2 :8
Table of 2 :10
Table of 2 :12
Table of 2 :14
Table of 2 :16
Table of 2 :18
Table of 2 :20
```

Example on (while loop)

```
package loops;

public class WhileLoopExample
{
public static void main(String[] args)
    {
        int i=1;    // start condition or Starting value
        while(i<=5)
        {
            System.out.println(i);  // End Condition
            i++;    // Increment or Decrement
        }
    }
}
```

```
<termin
1
2
3
4
5
```

WhileLoopExample.java | *WhileLoopEx2.java

```java
package loops;

public class WhileLoopEx2
{
    public static void main(String[] args)
    {
    int i=100;      //Start Condition
    while(i>=1)     //End Condition
        {
        System.out.println("Descending Order: "+i);
        i=i-2;      // Increment or Decrement
        }
    }

}
```

Console

```
<terminated> WhileLoopEx2 [Java Ap
Descending Order: 100
Descending Order: 98
Descending Order: 96
Descending Order: 94
Descending Order: 92
Descending Order: 90
Descending Order: 88
Descending Order: 86
Descending Order: 84
Descending Order: 82
Descending Order: 80
Descending Order: 78
Descending Order: 76
Descending Order: 74
Descending Order: 72
```

Example on (Do while loop)

*DoWhileLoop.java

```java
package loops;

public class DoWhileLoop
{
    public static void main(String[] args)
    {
        int i=10;   //Start Condition
        do
        {
            System.out.println(i);
            i++;    // Increment or Decrement
        }
        while (i<=15); // End Condition
    }
}
// Note 1: In Do While Loop it will always print 1st or Start Condition
// whether remaining condition are true or false.
// Note 2: If After executing first condition if remaining conditions are
// true then it will print all values otherwise only First Condition execute.
```

Console

```
<terminated> DoWhileLoop [Java App
10
11
12
13
14
15
```

WhileLoopEx... | DoWhileLoop.... | *DoWhileLoo...

```java
package loops;

public class DoWhileLoopEx2
{
    public static void main(String[] args)
    {
        int i=100;      //Start Condition
        do
        {
            System.out.println(i);
            i--;        //Increment or Decrement
        }
        while(i>=50);   //End Condition
    }
}
```

```
<termin
100
98
96
94
92
90
88
86
84
82
80
78
76
74
72
```

Control Statements

A) Selection Statement
B) Iteration Statement
C) Jump Statement

(1. if 2. If else 3. else if 4.nested if 5. Switch)

---

1. **If statement**
   (Will print only if the condition is true. If condition is false then print nothing)

```
package Control_Statements;

public class example1_IF_Statement
{
    public static void main(String[] args) {

            int marks= 25;

            // 25>=35
            if(marks>=35)
            {
                    System.out.println("Pass");
            }
    }

}
```

```
*If_Statement.java
1  package loops;
2
3  public class If_Statement
4  {
5      public static void main(String[] args)
6      {
7      int marks = 45;
8      if(marks >=35)          // Condition
9      {
10         System.out.println("Pass");
11     }
12
13     }
14
15 }
16 // It will print only when the result is true.
```

```
Cons
<terminat
Pass
```

---

2. If Else Statement

```
package Control_Statements;

public class example2_IF_Else_Statement
{
public static void main(String[] args)
        int marks = 34;
        if(marks>=35)
        {
                System.out.println("Pass");
        }
        else
        {
                System.out.println("Fail");
        }
}
}
```

```
*If_Else1.java
1  package control_Statement;
2
3  public class If_Else1
4  {
5      public static void main(String[] args)
6      {
7          int Age = 21;
8
9          if(Age>=18)
10         {
11             System.out.println("Eligible for Voting");
12         }
13
14         else
15         {
16             System.out.println("He is still a Kid");
17         }
18     }
19 }
20
```

```
Console
<terminated> If_Else1 (1) [Java Ap
Eligible for Voting
```

```
Outline
control_Statement
If_Else1
```

```
If_Else1.java
1  package control_Statement;
2
3  public class If_Else1
4  {
5      public static void main(String[] args)
6      {
7          int Age = 15;
8
9          if(Age>=18)
10         {
11             System.out.println("Eligible for Voting");
12         }
13
14         else
15         {
16             System.out.println("He is still a Kid");
17         }
18     }
19 }
```

```
Console
<terminated> If_Else1 (1) [Java
He is still a Kid
```

```
Outline
control_Statement
```

3. Else if Statement

```
package Control_Statements;
public class example3_Else_IF {
public static void main(String[] args) {
        int marks = 25;
        if(marks>=65)           //25>=65
        {
                System.out.println("Distinction");
        }
        else if (marks>=60 )            //25>=60
        {
                System.out.println("1st class");
        }
        else if (marks>=55)             // 25>=55
        {
                System.out.println("higher 2nd class");
        }
        else if (marks>=50)             //25>=50
        {
                System.out.println("2nd class");
        }
        else if (marks>=35)             //25>=35
        {
                System.out.println("Pass");
        }
        else
        {
                System.out.println("fail");
        }
```

Else_If1.java

```
1  package control_Statement;
2  public class Else_If1
3  {public static void main(String[] args)
4  {
5      int Distance = 75;
6      if (Distance>=100)
7      {
8          System.out.println(Distance+" KM is a Long Distance");
9      }
10
11     else if (Distance>=70)
12     {
13         System.out.println(Distance+" KM is a Medium Distance");
14     }
15
16     else if (Distance>=40)
17     {
18         System.out.println(Distance+" KM is a Short Distance");
19     }
20
21     else
22     {
23         System.out.println("Keep Walking");
24     }
25 }
26 }
```

Console

```
<terminated> Else_If1 [Java Application] C:\P
75 KM is a Medium Distance
```

4. Nested If

```
package Control_Statements;
public class example4_nested_if {
	public static void main(String[] args) {
		String UN="abc";
		String PWD ="xyz";
		if("abc"==UN)	//outer if
		{
			System.out.println("Correct UN: ");
			if("xyz"==PWD)	//inner if or nested if
			{
				System.out.println("Correct PWD: Login successfull");
			}
			else
			{
				System.out.println("Wrong PWD--> Login Failed");
			}
		}
		else
		{
			System.out.println("wrong UN--> Login failed");
		}
	}
	}
```

*Nested_If1.java

```java
package control_Statement;

public class Nested_If1
{
    public static void main(String[] args)
    {
        String UID = "abc";
        String PASS= "Vaibhav@123";
        if (UID =="abc") // Outer If
        {
            System.out.println("Correct UID");
            if (PASS=="Vaibhav@123") // Inner or Nested IF
                {
                    System.out.println("Correct PASS");
                }
            else
            {
                System.out.println("Wrong PASS");
            }
        }
        else
        {
            System.out.println("Wrong UID");
        }
    }
}
```

Console

```
<terminated> Nested_If1 [Java Ap
Correct UID
Correct PASS
```

5. Switch

```
package Control_Statements;
public class example5_switch1 {
public static void main(String[] args) {
        int inp=8;
        switch (inp)
        {
        case 1:  System.out.println("Today is mon: "); // 1 to 7 are the int inputs
        break;

        case 2:  System.out.println("Today is tue: ");
        break;
        case 3:  System.out.println("Today is wed: ");
        break;
        case 4:  System.out.println("Today is thr: ");
        break;
        case 5:  System.out.println("Today is fri: ");
        break;
        case 6:  System.out.println("Today is sat: ");
        break;
        case 7:  System.out.println("Today is sun: ");
        break;

        default:  System.out.println("wrong inputs");
        }
}
}
```

Switch_1.java Switch_2.java

```
1  package control_Statement;
2  public class Switch_2
3  {public static void main(String[] args) {
4      int inp=11;
5      switch (inp) {
6      case 1: System.out.println("Today is Sunday");
7      break;
8
9      case 2: System.out.println("Today is Monday");
10     break;
11
12     case 3: System.out.println("Today is Tuesday");
13     break;
14     case 4: System.out.println("Today is Wednesday");
15     break;
16     case 5: System.out.println("Today is Thurday");
17     break;
18     case 6: System.out.println("Today is Friday");
19     break;
20     case 7: System.out.println("Today is Saturday");
21     break;
22
23     default:System.out.println("Enjoy the Holiday");
24         break;
25     }
26     }
27 }
```

Console
<terminated> Switch_2 [Java Ap
Enjoy the Holiday

• Keywords and Identifiers

Java Keywords:

Keywords are predefined, reserved words used in Java programming that have special meanings to the compiler. For example:

abstract	assert	boolean	break	byte	case
catch	char	class	const	continue	default
do	double	else	enum	extends	final
finally	floatfor	goto	if	implements	import
instanceof	int	interface	long	native	new
package	private	protected	public	return short	static
strictfp	super	switch	synchronized	this	throw
throws	transient	try	void	volatile	while

Java identifiers

Identifiers are the name given to variables, classes, methods, package etc

Rules for Naming an Identifier

1. **Identifiers cannot be a keyword.**
2. **Identifiers are case-sensitive.**
3. **It can have a sequence of letters and digits. However, it must begin with a letter, $ or _. The first letter of an identifier cannot be a digit.**
4. **It's a convention to start an identifier with a letter rather and $ or _.**
5. **Whitespaces are not allowed. Similarly, you cannot use symbols such as @, #, and so on.**

eg.

```
// int static, String for, class while
// s1, m1, str2, sample1
// abc1, $ab1, _s1,
// 1abc, 2s, 5c1,
// s 1,  str 2,  abc 5,
// abc@1, str#2
```

• Types of variables

1. Local variable:

- **The variable which is declared inside the method/block/constructor is called local variable.**
- **Scope of local variable remains only within the method/block/constructor.**
- **Local variable is also called temporary variable.**

2. Global variable:

- **The variable which is declared outside the method/block/constructor is called global variable.**
- **Scope of global variable remains through the class.**
- **Global variable is also called permanent variable.**

3. Static/Class variable:

1. static variable call from same class --> variableName();
2. static variable call from diff class --> className.variableName();

Note: we can access static global variable in both static & non-static method

4. Non-static / Instance variable: (instance-object)

3. non-static variable call from same class
4. non-static variable call from diff class

Example1: Local & Global Variable

```
package Types_Of_variables;
public class sample1
{
	int a=10; //global variable
	public void m1()
	{
		int b=20;		//local variable
		System.out.println(b);	//20
		System.out.println(a);	//10
	}
	public void m2()
	{
		int c=30;		// local variable
		System.out.println(c);	//30

		//System.out.println(b);
		System.out.println(a);	//10
	}
	public static void main(String[] args) {
		sample1 s1=new sample1();
		s1.m1();
		s1.m2();
	}
}
```

Example2: Static Global Variable Call from Same Class & Different Class

```
package Types_Of_variables;

public class sample2
{


        static int a=10;                                //static global variable from same class

        public static void main(String[] args)
        {
                //1. static global variable call from same class
                System.out.println(a);                  //10 // variableName

                //2. static global variable call from diff class
                System.out.println(sample3.b);          //20 //classname.variablename

        }

        public static void m1()
        {
                System.out.println(a);
        }


        public void m2()
        {
                System.out.println(a);
        }

}
```

```
package Types_Of_variables;

public class sample3 {


        static int b=20;                //static global variable from diff class


        public static void m2()
        {

        }

}
```

Example3: Non-static Global Variable Call from Same Class & Different Class

```
package Types_Of_variables;

public class sample4
{

        int a=40;                       //non-static global variable from same class

        public static void main(String[] args) {

                //3. non-static global variable call from same class
                sample4 s4= new sample4();          // create object of same class or current
                System.out.println(s4.a);           // objectName.variable

                //4. non-static global variable call from diff class
                sample5 s5 =new sample5();          // create object of diff class
                System.out.println(s5.b);           // objectName.variable

        }

        public void m3()
        {
                System.out.println(a);
        }

        public static void m4()
        {
                //System.out.println(a);
        }

}
```

```
package Types_Of_variables;

public class sample5 {

        int b=50;   //non-static global variable diff class


        public void m5()
        {
                System.out.println(b);
        }

}
```

Variables.java

```java
package Types_of_Variables;
public class Variables
{
        private static Variables s1;
        int a = 10;                 // int a is declared globally
    public void m1()

    {                               // int b declared inside the method
        int b = 20;                 // so int b is local variable
        System.out.println("Print thealue of b: "+b);
        System.out.println("Print thealue of a: "+a);
    }
    public void m2()
    {
        int c = 30;
        System.out.println("Print thealue of c: "+c);
        System.out.println("Print thealue of a: "+a);
    }
    public static void main(String[] args)
    {
        Variables.s1 = new Variables(); // Classname.ObjectName = new Classname();
        s1.m1();        // create object bcoz Non Static Method Used
        s1.m2();
        System.out.println(s1.a);
    }
}
```

Console

```
<terminated> Variables [Java Application]
Print thealue of b: 20
Print thealue of a: 10
Print thealue of c: 30
Print thealue of a: 10
10
```

Static Global Variable Call from Same Class & Different Class

Static_Global_Variable.java | *Static_Global_Variable_Another_Class.java

```java
package Types_of_Variables;

public class Static_Global_Variable
{
    static int a = 10;

    public static void main(String[]args)
    {
    //1. Static global variable call from same class
        System.out.println(a);

    //2. static global variable call from different class
        System.out.println(Static_Global_Variable_Another_Class.b);
    }                           //classname.VariableName
    public static void m1()
    {
        System.out.println(a);
    }
    public void m2()
    {
        System.out.println(a);
    }

}
```

Console

```
<terminated> Static_Global_Variabl
10
20
```

Class :1

Variables.java | Static_Global_Variable.java | *Static_Global_Variable_Another_Class.java

```java
package Types_of_Variables;

public class Static_Global_Variable_Another_Class
{
    static int b = 20;          // Static Global Variable
    public static void m2()
    {

    }
}
```

Class : 2

Non-static Global Variable Call from Same Class & Different Class

NonStatic_Global_Variable.java | NonStatic_Global_Another_C.java

Class : 1

```java
package Types_of_Variables;

public class NonStatic_Global_Variable
{

    int a = 10;     //Non-Static Global Variable from Same Class
    public static void main(String[] args)
    {
        //3. Non-Static Global Variable Call from Same Class
        NonStatic_Global_Variable s1 = new NonStatic_Global_Variable();
                                        // Create Object of Current Class
        System.out.println(s1.a);       //ObjectName.VariableName

        //4. Non-Static Global Variable Call from Different Class
        NonStatic_Global_Another_C s2 = new NonStatic_Global_Another_C();
                                // Create Object From Different Class
        System.out.println(s2.b);   //ObjectName.VariableName
    }

```

Console

```
<terminated> N
10
50
```

NonStatic_Global_Variable.java | NonStatic_Global_Another_C.java

```java
package Types_of_Variables;

public class NonStatic_Global_Another_C
{
    public static NonStatic_Global_Another_C o2;
    int b=50;   //non-static global variable Different class

    public void m5()
    {
        System.out.println(b);
    }

}
```

Class : 2

Demo2.java | Demo1.java

```java
package Types_of_Variables;

public class Demo2
{
    int a = 10;             //non-static global variable from same class
    public static void main(String[] args)
    {
        //3. non-static global variable call from same class
        Demo2 d1 = new Demo2();     // create object of same class or current
        System.out.println(d1.a);   // objectName.variable

        //4. non-static global variable call from diff class
        Demo1 d2 = new Demo1();     // create object of diff class
        System.out.println(d2.b);   // objectName.variable

    }
    public void m3()
    {
    }
    }
```

Console

```
<terminated> Dem
10
20
```

Demo2.java | *Demo1.java

```java
package Types_of_Variables;

public class Demo1
{
    int b=20;               //non-static global variable diff class
    public void m5()
    {
        System.out.println(b);
    }
}
```

- **Constructor:**

A constructor in Java is a special method that is used to initialize objects/variables.
The constructor is called when an object of a class is created.

At the time of constructor declaration below points need to follow:

1. **Constructor name should be same as class name**
2. **you should not declare any return type for the constrictor (like void).**
3. **Any no of constructor can be declared in a java class but constructor name should be same as class name, but arguments/parameter should be different.**

Use of Constructor

1. **To copy/load all members of class into object --> when we create object of class**
2. **To initialize data member/variable**

- **Types of Constructor**

1. **Default Constructor**
2. **User defined Constructor**

1. **Default Constructor**

- **If Constructor is not declared in java class, then at the time of compilation compiler will provide Constructor for the class**
- **If programmer has declared the constructor in the class, then compiler will not provide default Constructor.**
- **The Constructor provided by compiler at the time of compilation is known as Default Constructor**

2. **User defined Constructor**

- **If programmer is declaring constructor in java class, then it is considered to be as User defined constructor.**

User defined Constructor are classified into 2 types

1. **Without/zero parameter constructor**

   // Example-without parameter constructor --e.g., addition

2. **With parameter constructor**

   // Example-with parameter constructor- only 1 constructor -- e.g., addition

   // Example-with parameter constructor- multiple constructor -- e.g., addition

---

Example1: default constructor (Class1)

```
package Constructor;
public class sample1 {
	//	default constructor---> provided by compiler
	//	use1: to copy all the members of class into object --> after object creation
	//	sample1() {		}
	public static void main(String[] args)
	{
		sample2 s2=new sample2();
		s2.m1();
		//1 sample2 --> classname -->datatype
		//2 s2  --> objectName --> to identify object
		//3 new -->keyword --> to create blank/empty object
		//4 sample2() --> classname() --> constructor

		sample1 s1=new sample1();		//Object Creation
		s1.m2();
	}
	//Non-static Regular Method
	public void m2()
	{
		System.out.println("Running method m2 from same class");
	}
}
```

Example1: default constructor (Class2)

```
package Constructor;
public class sample2
{
	//	sample2() {		}
	//non-static regular method
	public void m1()
	{
		System.out.println("Running method m1 from diff class");
	}
}
```

Example2: User defined constructor (Class 1)

```
package Constructor;

public class sample3
{
        //Step1: variable declaration
        int num3;       // 30
        int num4;       //5
                        //user defined --> provided by user
                        //use1: to initialize global variable
                        //use2: to copy all the members of class into object

        //Step2: variable initialization
        sample3()
        {
                num3 = 30;
                num4 = 5;
        }

        //Step3: usage
        public void div()
        {
                int divValue = num3/num4;           //30/5  = 6
                System.out.println(divValue);
        }

public static void main(String[] args) {

        sample4 s4=new sample4();
        s4.addition();
        s4.mul();

        System.out.println("---------------");

        sample3 s3=new sample3();
        s3.div();
        }
}
```

Example2: User defined constructor (Class 2)

```
package Constructor;
public class sample4
{
        //Step1: variable declaration
        int num1;       //10
        int num2;       //20

                //user defined constructor --> provided by user
                //use1: to initialize global variables/object
                //use2: to copy all the members of class into object



//Step2: variable initialization
        sample4()
        {
                num1 = 10;
                num2 = 20;
        }

        //Step3: usage
        public void addition()
        {
                int sum = num1+num2;
                System.out.println(sum);
        }

        //Step3: usage
        public void mul()
        {
                int mulValue = num1*num2;
                System.out.println(mulValue);
        }
}
```

---

*UserDefConsSC.java | UserDefConAnother.java

```java
package Constructor;                    //user defined --> provided by user

public class UserDefConsSC              //use1: to initialize global variable
{                                       //use2: to copy all the members of /n
//Step1: variable declaration           //class into object
int speed;
int time;

//Step2: variable initialization        // Create User Constructor
UserDefConsSC()     // Name of Constructor and ClassName Must be Same
    {
        speed = 40;
        time = 10;
    }

//Step3: usage
public void distance()
    {
        int disValue = speed*time;
        System.out.println(disValue);
    }

public static void main(String[] args)
{
    UserDefConsSC o1 = new UserDefConsSC(); //Object Creation
        o1.distance();                      //objectName.MethodName

        UserDefConAnother Another1 = new UserDefConAnother();
        Another1.velocity();
        Another1.addition();

}
}
```

Console

```
<terminated> Us
400
12.575
543.0
```

*UserDefConsSC.java | *UserDefConAnother.java

```java
package Constructor;
public class UserDefConAnother
{   //Step1: variable declaration
    float displacement;
    float timeInMin;
        //Step2: variable initialization    // Create User Constructor
    UserDefConAnother() // Name of Constructor and ClassName Must be Same
    {
        displacement = 503;
        timeInMin    = 40;
    }
        //Step3: usage
    public void velocity()
    {
        float vCity = displacement/timeInMin;
        System.out.println(vCity);
    }
    //Step3: usage
        public void addition()
        {
            float add = displacement+timeInMin;
            System.out.println(add);
        }
}
```

Class : 2

Example of with parameter constructor (1 constructor) (Class:1)

```
package Constructor;
public class sample5
{
        int num3;       //50
        int num4;       //5

        //      user defined --> with parameter constructor (int, int parameter)
        //      use1: to initialize global variable
        //      use2: to copy all members of class into object
        sample5(int c, int d)
        {
                num3=c;                 // assign local variable info into global variable
                num4=d;
        }
        public void div()
        {
                int divValue = num3/num4;           //50/5  ==10
                System.out.println(divValue);
        }
        public static void main(String[] args)
        {
                sample6 s6=new sample6(50,10);
                s6.addition();
                s6.mul();

                sample5 s5=new sample5(100, 5);
                s5.div();
        }
}
```

Example of with parameter constructor (1 constructor) (Class:2)

```
package Constructor;
public class sample6
{
        int num1;       //20
        int num2;       //25

        sample6(int a, int b)
        {
                num1=a;         //20    //assign local variable info into global variable
                num2=b;         //25
        }
        public void addition()
        {
                int sum= num1 + num2;
                System.out.println(sum);
        }
        public void mul()
        {
                int mulValue= num1 * num2;
                System.out.println(mulValue);
        }
}
```

*WithParameter1.java WithParameter2.java

```java
package Constructor;
public class WithParameter1
{
    int num1;                       //Step1: variable declaration
    int num2;

    WithParameter1(int a, int b)    //Step2:variable initialization
    {
        num1 =a;
        num2 =b;
    }

    public void addition()
    {
        int addValue = num1+num2;
        System.out.println("Add Value = "+addValue);
    }

    public void subtract()
    {
        int subValue = num1-num2;
        System.out.println("Sub Value = "+subValue);
    }

    public static void main(String[] args)
    {
        WithParameter1 o1 = new WithParameter1(60,80);
                o1.addition();
                o1.subtract();

        WithParameter2 a1 = new WithParameter2(20,5);
                a1.multiply();
                a1.division();

        WithParameter2 a2 = new WithParameter2("Vaibhav");
                a2.StudentName();
    }
}
```

```
<terminated> WithParameter1 [Java Appl
Add Value = 140
Sub Value = -20
Mul Value = 100
Div Value = 4
Student Name = Vaibhav
```

Class :1

*WithParameter1.java WithParameter2.java

```java
package Constructor;
public class WithParameter2
{
        int num3;                       //Step1: variable declaration
        int num4;
        String sName;

        WithParameter2(int c, int d)    //Step2:variable initialization
        {
            num3 = c;
            num4 = d;
        }

        WithParameter2(String str)  //Step2:variable initialization
        {
            sName = str;
        }

        public void multiply()          //Step3 : Usage
        {
            int mulValue = num3*num4;
            System.out.println("Mul Value = "+mulValue);
        }

        public void division()
        {
            int divValue = num3/num4;
            System.out.println("Div Value = "+divValue);
        }

        public void StudentName()
        {
            String Student = sName;
            System.out.println("Student Name = "+Student);
        }
}
```

Class :2

Example of With Parameter Constructor (Multiple Constructor) (Class:1)

```
package Constructor;
public class sample7
{

        public static void main(String[] args)
         {

                sample8 s8=new sample8(10, 20);
                s8.addition();

                sample8 s9=new sample8("Atul");
                s9.studentName();
        }

}
```

Example of With Parameter Constructor (Multiple Constructor) (Class:2)

```
package Constructor;
public class sample8
{

        int num1;               //10
        int num2;               //20
        String sname;           //Rahul

        //user defined -->with parameter --> (int, int parameter constructor)
        sample8(int a, int b)
        {
                num1=a;                 //10
                num2=b;                 //20
        }


        //user defined -->with parameter --> (String parameter constructor)
        sample8(String str)
        {
                sname = str;            //Rahul
        }

        public void studentName()
        {
                System.out.println(sname);
        }

        public void addition()
        {
                int sum = num1 + num2;
                System.out.println(sum);
        }

}----------------------------------------------------------------------------------------------------------------
```

```
package Constructor;
public class sample9
 {
        public static void main(String[] args)
        {
                sample10 s10=new sample10("rahul", 100, 'A', 65.5f);
                s10.studentInfo();      // objectName.methodName();
        }
}
```

```
package Constructor;
public class sample10
{
        String StudentName;          // Global Variable
        int StudentRollNum;
        char StudentGrade;
        float StudentPer;
        sample10(String sname, int srollnum, char sgrade, float sper)    // User Constructor
        {
                StudentName = sname;
                StudentRollNum = srollnum;
                StudentGrade = sgrade;
                StudentPer = sper;
        }
        public void studentInfo()
        {
                System.out.println(StudentName);
                System.out.println(StudentRollNum);
                System.out.println(StudentGrade);
                System.out.println(StudentPer);
        }
}
```

ZSample1.java | ZSample2.java

```java
package Constructor;

public class ZSample2
{
    String StudentName;
    int StudentRollNum;
    char StudentGrade;
    float StudentPercent;

    ZSample2(String SName, int SRoll,char SGrade, float SPer  )
    {
        StudentName = SName;
        StudentRollNum = SRoll;
        StudentGrade=SGrade;
        StudentPercent=SPer;
    }

    public void studentinfo()
    {
        System.out.println(StudentName);
        System.out.println(StudentRollNum);
        System.out.println(StudentGrade);
        System.out.println(StudentPercent);
    }
}
```

Console

```
<terminated> ZSampl
Vaibhav
23
A
95.5
```

ZSample1.java | ZSample2.java

```java
package Constructor;

public class ZSample1
{
    public static void main(String[]args)
    {
        ZSample2 Z2 = new ZSample2("Vaibhav", 23, 'A',95.5f);


        Z2.studentinfo();
    }
}
```

Use of Static and Non-Static Variables

```
package Static_non_Static_Use;

public class emp
{
        int eid;                        //non-static global variable
        int esal;
        static  String eceoname;

        public void showInfo()
        {
                System.out.println(eid+" :"+esal+ " :"+eceoname);
        }
}
```

```
package Static_non_Static_Use;
public class static_use
{
        public static void main(String[] args)
        {

                emp rahul= new emp();
                rahul.eid = 100;
                rahul.esal = 10000;
                emp.eceoname  = "xyz";

                emp nikhil =new emp();
                nikhil.eid=200;
                nikhil.esal=20000;
                emp.eceoname="lmn";

                emp shree=new emp();
                shree.eid=300;
                shree.esal=30000;
                emp.eceoname="abcd";

                System.out.println("----info of emp rahul-------");
                rahul.showInfo();

                System.out.println("----info of nikhil rahul-------");
                nikhil.showInfo();

                System.out.println("----info of shree rahul-------");
                shree.showInfo();
        }

}
```

Impact of Using Static and Non Static Variables

*zEmployee.java | ZEmployeeUSE.java

```java
package Constructor;

public class ZEmployeeUSE
{
    public static void main(String[] args)
    {
        zEmployee e1 = new zEmployee();
        e1.empID = 10;
        e1.empName = "Anand";
        e1.CEOName = "Harshada";

        zEmployee e2 = new zEmployee();
        e2.empID = 20;
        e2.empName = "Rohan";
        e2.CEOName = "Anil";

        e1.showInfo();      // ObjectName.ClassName();
        e2.showInfo();      // ObjectName.ClassName();

    }
}
```

*zEmployee.java | ZEmployeeUSE.java

```java
package Constructor;

public class zEmployee
{
 int empID;
 String empName;
 static String CEOName;     // Using Static

 public void showInfo()
 {
     System.out.println(empID+":"+empName+"--->"+CEOName);
 }
}
```

Console

```
<terminated> ZEmployeeUSE [Java
10:Anand--->Anil
20:Rohan--->Anil
```

zEmployee.java | ZEmployeeUSE.java

```java
package Constructor;

public class zEmployee
{
 int empID;
 String empName;
 String CEOName;        // Using Non Static

 public void showInfo()
 {
     System.out.println(empID+":"+empName+"--->"+CEOName);
 }
}
```

Console

```
<terminated> ZEmployeeUSE [Java A
10:Anand--->Harshada
20:Rohan--->Anil
```

Example on Access Variables

```
package Sample1;
public class NotepadPlus
{
      int a = 10;
      static int b = 20;
      public static void m1()
{
int a = 30;
System.out.println(a);
}
      public void m2()
      {
      int a = 40;
      System.out.println(a);
      }

public static void main(String[] args)
{
int a = 50;

      NotepadPlus n1 = new NotepadPlus();
      System.out.println(n1.a);                //10
      System.out.println(b);                   //20
      NotepadPlus.m1();                        //30
      n1.m2();                                 //40
      System.out.println(a);                   //50

NotepadAnother n2 = new NotepadAnother();      //60
System.out.println(n2.a);                      //70
System.out.println(NotepadAnother.b);
NotepadAnother.m3();                           //80
n2.m4();                                       //90
}
}
```

```
package Sample1;
public class NotepadAnother
{
int a = 60;
static int b = 70;
public static void m3()
      {
      int a = 80;
      System.out.println(a);
      }

public void m4()
{
int a = 90;
System.out.println(a);
}
}
```