INDEX :

Custom Class :

```java
class Employee{

    int id;

    String name;

    public void printDetails(){

        System.out.println("My id is : " + id);

        System.out.println( name);

    }

}

public class M_CustomClass{

    public static void main(String[] args) {

        System.out.println("This is our custom class :-");

        Employee itsMe = new Employee(); // Instantiating  a new Employee 'object'.

        itsMe.id = 101; // Setting Attributes

        itsMe.name = "I'm here";

        System.out.println(itsMe.id);

        System.out.println(itsMe.name);

        itsMe.printDetails();

        Employee ItsMe = new Employee();

        ItsMe.id = 12;

        ItsMe.name = "Where are you ?";

        ItsMe.printDetails();


    }

}
```

Output :-

```
PS D:\11. Tutorial of Java> cd
This is our custom class :-
101
I'm here
My id is : 101
I'm here
My id is : 12
Where are you ?
PS D:\11. Tutorial of Java>
```

Constructor :

```
class Employee{

    private int id;

    private String name;

    Employee(int i, String n){

        id = i;

        name = n;

    }

    public void getId(){

        System.out.println("Id of the 'Employee' is : " + id);

    }

    public void getName(){

        System.out.println("Name of 'Employee' is : " + name);

    }

}


public class O_Constructor{

    public static void main(String[] args){

        Employee itsMe = new Employee(102, "Aman Verma");

        itsMe.getId();

        itsMe.getName();

    }

}
```

Inheritence :

```
class Base{
    private int x;
    public void setX(int x){
        this.x = x; /* 'this' is a way for us to refere an object of the class which is
        being created/referenced.
        'this' is a reference to current object.
        */
    }
    public int getX(){
        System.out.println("I'm in base and setting 'x' now");
        return x;
    }
}

class Derived extends Base{
    private int y;
    public void setY(int y){
        this.y = y;
    }
    public int getY(){
        System.out.println("I'm in derived and setting 'y' now");
        return y;
    }
}

public class P_Inheritence{
    public static void main(String[] args) {
        Derived d = new Derived();
        d.setX(12);
        d.setY(21);
        System.out.println(d.getX());
        System.out.println(d.getY());
    }
}
```

Output :-

```
PS D:\11. Tutorial of Java> cd "d:\11.
I'm in base and setting 'x' now
12
I'm in derived and setting 'y' now
21
PS D:\11. Tutorial of Java>
```

Constructor In Inheritence :

```
class Base{
    Base(){
        System.out.println("I'm a constructor of base class");
    }

    Base(int a){
        System.out.print("I'm an overloaded constructor of base class ");
        System.out.println("with value of a is : " + a);
    }
}

class Derived extends Base{
    Derived(){
        super(21); /* --> This will pass the value to the parameterised constructor of
        the 'Base' class.
        'super' keyword is a reference variable used to refer immediate parent class object
        */
        System.out.println("I'm a constructor of derived class");
    }

    Derived(int a, int n){
        super(a);
        System.out.print("I'm an overloaded constructor of base class ");
        System.out.println("with value of n is : " + n);
    }
    public void explaination(){
        System.out.println("The constructor of the 'Base' class is executed first");
        System.out.println("And followed by the constructor of the 'Derived' class");
    }
}

public class Q_ConstructorInInheritence{
    public static void main(String[] args) {
        Derived d = new Derived(12, 21);
```

```
        /*
        When a derived class is extended from the base class, the constructor of the 'Base'
        class is executed first followed by the constructor of the 'Derived' class.
        In case if base class have an overloaded constructor then still unparameterised
        constructor will called.
        */
        d.explaination();
    }
}
```

Output :-

```
PS D:\11. Tutorial of Java> cd "d:\11. Tutorial of Java\" ; if ($?)
I'm an overloaded constructor of base class with value of a is : 12
I'm an overloaded constructor of base class with value of n is : 21
The constructor of the 'Base' class is executed first
And followed by the constructor of the 'Derived' class
PS D:\11. Tutorial of Java>
```

Method Overriding :

```java
class A{
    public int method1(){
        return 12;
    }
    public void method2(){
        System.out.println("I'm 2nd method of class A");
    }
}

class B extends A{
    @Override // This will override the method of class 'A'
    public void method2(){
        System.out.println("I'm 2nd method of class B");
    }
    public void method3(){
        System.out.println("I'm 3rd method of class B");
    }
}
public class U_MethodOverriding{
    public static void main(String[] args) {
        A a = new A();
        a.method2();

        B b = new B();
        System.out.println(b.method1());
        b.method2(); // This will override the 'method2' of class 'A'
    }
}
```

Output :-

```
PS D:\11. Tutorial of Java> cd "d:\1
I'm 2nd method of class A
12
I'm 2nd method of class B
PS D:\11. Tutorial of Java>
```

Dynamic Method Dispatch :

```java
class Phone{

    public void On(){

        System.out.println("Turning on Nokia 1800");

    }

    public void Greet(){

        System.out.println("Good Morning..!");

    }

}

class SmartPhone extends Phone{

    public void Welcome(){

        System.out.println("You are welcome..!");

    }

    public void On(){

        System.out.println("Turning on Vivo X70 Pro+");

    }

}


public class V_DynamicMethodDispatch{

    public static void main(String[] args) {

        Phone obj1 = new SmartPhone();

        // SmartPhone obj2 = new Phone(); // This will throw error

        obj1.Greet();

        obj1.On(); // This will override the method of class Phone.

        // obj1.Welcome(); // This will also throw error to run 'Welcome' method, we have to

        // make the object of SmartPhone.

        SmartPhone obj2 = new SmartPhone();

        obj2.Welcome();

    }

}
```

Output :-

```
PS D:\11. Tutorial of Java> cd "d:\11.
Good Morning..!
Turning on Vivo X70 Pro+
You are welcome..!
PS D:\11. Tutorial of Java>
```

Abstract Class & Methods :

```java
abstract class Base{

    // We cann't make object of an abstract class.

    Base(){

        System.out.println("I'm a constructor of the Base class");

    }

    public void Hey(){

        System.out.println("Hey I'm here..!");

    }

    abstract public void Greet();

}

class Derived extends Base{

    @Override

    public void Greet(){

        System.out.println("Good Morning..!");

    }

    /*  All the abstract methods should must be implemented in Derived Class otherwise it throw

    error or we have to make Derived class as an abstract class.  */

}

public class W_AbstractClassAndMethods{

    /*  Abstract Methods :- A method that is declared without an implementation.

    Ex :  ->  abstract void moveTo(double x, double y);


    Abstract Class :- If a class includes abstract methods, then the class itself must be

    declaired abstract, as in.  */

    public static void main(String[] args) {

        // Base b = new Base(); // This will throw error as 'Base' is an abstract class.

        Derived d = new Derived();

        d.Greet();

    }

}
```

Output :-

```
PS D:\11. Tutorial of Java> cd "d:\11
I'm a constructor of the Base class
Good Morning..!
PS D:\11. Tutorial of Java>
```