```
//Method Demonstration
class Addition {
   int sum = 0;
   public int addTwoInteger(int a, int b){
      sum = a + b;
      return sum;
   }
}
class Main {
   public static void main (String args []) {
      //Creating an object and memory allocation
      Addition add = new Addition();
      //calling the method
      int s = add.addTwoInteger(20, 10);
      System.out.println("The sum of two integer = " + s);
   }
}
```

```
//Nested Class
class OuterClass {
   int x = 10;
   class InnerClass {
      int y = 5;
   }
}
class nested {
    public static void main (String args []) {
        //OBJ creation and memory allocation
        OuterClass myOuter = new OuterClass();
        OuterClass.InnerClass myInner = myOuter.new InnerClass();
        System.out.println(myInner.y + myOuter.x);
    }
}
```

//Constructor

```
class Demo {
    int value1;
    int value2;
    //Creating a constructor using the same name as a class
    Demo() {
        value1 = 10;
        value2 = 20;
        System.out.println("Inside Constructor");
    }
    //Creating a method
    public void display() {
        System.out.println("Value1 = " + value1);
        System.out.println("Value2 = " + value2);
    }
    public static void main (String args []) {
        //Creating an obj of and memory allocation
        Demo d = new Demo();
        d.display();
    }
}
```

//Method Overloading program

```
class Sum {
    public int sum (int a, int b){
        return (a + b);
    }
    public int sum (int a, int b, int c) {
        return (a + b + c);
    }
    public static void main(String args []) {
        //Creating an object and memory allocation
        Sum s = new Sum();
        System.out.println(s.sum(20,20));
        System.out.println(s.sum(10,30,40));
    }
}
```

//Constructor Overloading

```
class person {
   //Declaring a default constructor
   person () {
      System.out.println("Hello");
   }
   //Declaring a parameterized constructor
   person(String name) {
      System.out.println(name);
   }
   //Main Method
   public static void main(String args []) {
      //Creation an object and memory allocation
      person p1 = new person();
      person p2 = new person("Usman");
   }
}
```

//Method overriding

```
class animal {
   public void display() {
      System.out.println("I am animal");
   }
}
class tiger extends animal{
   public void display() {
      System.out.println("i am tiger");
   }
   public static void main(String args []) {
      tiger t1 = new tiger();
      t1.display();
   }
}
```

//INTERFACE

```
interface Animal {
   public void animalSound();
   public void sleep();
}
class pig implements Animal {
   public void animalSound(){
      System.out.println("The Pig Says: WEE-WEE");
   }
   public void sleep(){
      System.out.println("The Pig is sleeping");
   }
}
```

```
class interface1 {
   public static void main (String args []) {
      //Objection creation and memory allocation
      pig myPig = new pig();
      //Calling method
      myPig.animalSound();
      myPig.sleep();
   }
}
```

//Exception Handling

```
class except {
   public static void main(String args []) {
      try {
         int a = 30, b = 0, c;
         c = a / b;
         System.out.println("Result = " + c);
      }
      catch (ArithmeticException e) {
         System.out.println("Cannot devide by 0");
      }
   }
}
```

//Constructor Overloading

```
class Demo{
      int  value1;
      int  value2;
      Demo(){
       value1 = 10;
       value2 = 20;
       System.out.println("Inside 1st Constructor");
     }
     Demo(int a){
      value1 = a;
      System.out.println("Inside 2nd Constructor");
    }
    Demo(int a,int b){
    value1 = a;
    value2 = b;
    System.out.println("Inside 3rd Constructor");
   }
   public void display(){
      System.out.println("Value1 === "+value1);
      System.out.println("Value2 === "+value2);
  }
  public static void main(String args[]){
    Demo d1 = new Demo();
    Demo d2 = new Demo(30);

    Demo d3 = new Demo(30,40);

    d1.display();

    d2.display();

    d3.display();

 }

}
```