INTERITANCE:

Program1: Demo of inheritance

```
class  Demo  //total members in this class =3
{
   int a,b;
          void sum()
          {
          System.out.println("sum is:"+(a+b));
          }
}
class UseDemo extends Demo  //inheritance
{
    int c;
    void sumAll()
          {
          System.out.println("sumAll is:"+(a+b+c));
          }
}
class FirstProgram
{
  public static void main(String args[])
  {
    UseDemo d=new UseDemo();
           d.a=10;
           d.b=20;
           d.c=30;
           d.sum();
           d.sumAll();
  }
}
```

Program2: Same program using private variables

```
class  Demo
{
          private int a,b;      //private variables are not accessible outside this class
          void setValues(int a,int b)   //to initialized the value of private variable
          {
            this.a=a;
            this.b=b;
          }
          int sum()
          {
          System.out.println("sum is:"+(a+b));
          return a+b;
          }
}
class UseDemo extends Demo  //inheritance
{
   int c;
```

```
  void sumAll()
        {
        System.out.println("sumAll is:"+(sum()+c));
        }

}
class SecondProgram
{
 public static void main(String args[])
 {
  UseDemo d=new UseDemo();
        d.c=30;
        d.setValues(10,20);
        d.sumAll();

 }
}
```

Super Keyword:

First use of super in constructor calling.

Program3:First Use Of super in constructor calling

```
class  Demo  //total members in this class =3
{
  private int a,b;
        Demo(int a,int b)
        {
         this.a=a;
         this.b=b;
        }
        int sum()
        {
        System.out.println("sum is:"+(a+b));
        return a+b;
        }
}
class UseDemo extends Demo  //inheritance
{
   int c;
         UseDemo(int x,int y,int z)
         {
          super(x,y);
          c=z;
         }
   void sumAll()
         {
         System.out.println("sumAll is:"+(sum()+c));
         }
}
class ThirdProgram
{
```

```
  public static void main(String args[])
  {
    UseDemo d=new UseDemo(22,33,44);
          d.sumAll();
  }
}
```

Program 4:one more program of constructor calling using super keyword

```
class A
{
 A()
 {
 System.out.println("class A, cons called");
 }
}
class B extends A
{
 B()
 {
 System.out.println("class B cons called");
 }
}
class C extends B
{
 C()
 {
 System.out.println("class C ,cons called");
 }
}
class Main
{
 public static void main(String args[])
 {
  C oc=new C();
 }
}
```

Second Use Of Super:calling the hidden variable of super class due to the same name variable in the subclass:

Program5: Variable calling using super keyword

```
class  Demo
{
   int a=100;  //hidden
}
class UseDemo extends Demo
{
   int a=90;
```

```
        void show()
        {
        int a=50;
        System.out.println("value of a is:"+a);
        System.out.println("value of this.a is:"+this.a);
        System.out.println("value of super.a is:"+super.a);
        }
        void showMe()
        {
        System.out.println("Now a is:"+a);
        }
}
class SecondUseOfSuper
{
  public static void main(String args[])
  {
    UseDemo d=new UseDemo();
          d.show();d.showMe();
  }
}
```

Third Use Of Super: Method overrding

Program6: Method Overriding

```
//public-->protected--->default--->private
class  Demo
{
        public void show() //hidden
        {
        System.out.println("super class show method called");
        }
        void calc(double a,double b)
        {
        System.out.println("sum is:"+(a+b));
        }
}
class UseDemo extends Demo
{
       @Override          //annotation, it is optional
        public void show()
        {
        System.out.println("sub class's show method called");
        super.show();
        }

        @Override
        void calc(double a,double b)
        {
        System.out.println("power is:"+Math.pow(a,b));
        }
```

```
        void f1()
        {
         super.show();
        }
}
class ThirdUseOfSuper
{
  public static void main(String args[])
  {
    UseDemo d=new UseDemo();
        d.show();
        d.f1();
        d.calc(2,3);
  }
}
```

Final keyword:

Program7: First Use Of final keyword is to declare constants

```
class Demo
{
  final int num=100;
  void show()
  {
  num=110;  //error ,final variable can't be modified
  System.out.println("total students are:"+num);
  }
}
class FirstProgram
{
 public static void main(String args[])
 {
  Demo d=new Demo();
  d.num=-1;  //error ,final variable can't be modified
  d.show();
 }
}
```

Program8: Second Use OF final is to prevent class inheritance

```
final class Demo
{

}
class UseDemo extends Demo  //error because final class Demo can't be subclasses
{
}
class SecondProgram
{
```

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

 }
}
```

Program9:Third use of final is to prevent method overriding

```
class Demo
{
 final void authorName()
  {
  System.out.println("made by abhishek");
  }
}
class UseDemo extends Demo
{
  @Override
  void authorName()  //error overridden method is final
  {
  System.out.println("made by rahul!!");
  }
}
class ThirdProgram
{
 public static void main(String args[])
 {
  UseDemo d=new UseDemo();d.authorName();
 }
}
```

Abstract Class:

Program10: Method Overriding

```
abstract class Demo
{
 abstract void rollNo();
 abstract void getPower(double a,double b);
 abstract void name();
 void call()
 {
 System.out.println("call called");
 }
}
abstract class UseDemo extends Demo
{
 @Override
```

```
  void rollNo()
  {
   System.out.println("One");
   }

   @Override
   void getPower(double a,double b)
   {
   System.out.println(Math.pow(a,b));
   }
   void show()
   {
   System.out.println("show called");
   }
}

class UseDemo1 extends UseDemo
{
  @Override
  void name()
  {
  System.out.println("made by abhishek jain");
  }
}

class FirstProgram
{
  public static void main(String args[])
  {
   UseDemo1 d=new UseDemo1();
   d.show();d.getPower(3,4);d.rollNo();d.call();
  }
}
```

Assigning the object of sub class in the reference of super class:

Program10: Method Overriding

```
class Room
{
   void area()
   {
     System.out.println("area method called");
   }
}
class StudyRoom extends Room
{
  @Override
  void area()
  {
  System.out.println("area method in subclass called");
```

```
 }
 void volume()
 {
  System.out.println("volume method called");
 }
}
class UsingReferences
{
 public static void main(String args[])
 {
        // Room r=new StudyRoom();

          Room r;
          StudyRoom sr=new StudyRoom();
          r=sr;
          sr.area();
          r.volume();
 }
}
```