Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

SAIRAM DIGITAL RESOURCES

YEAR II SEM III

CS8392

OBJECT ORIENTED PROGRAMMING
(Common to CSE, EEE, EIE, ICE, IT)

UNIT NO 2

INHERITANCE AND INTERFACES

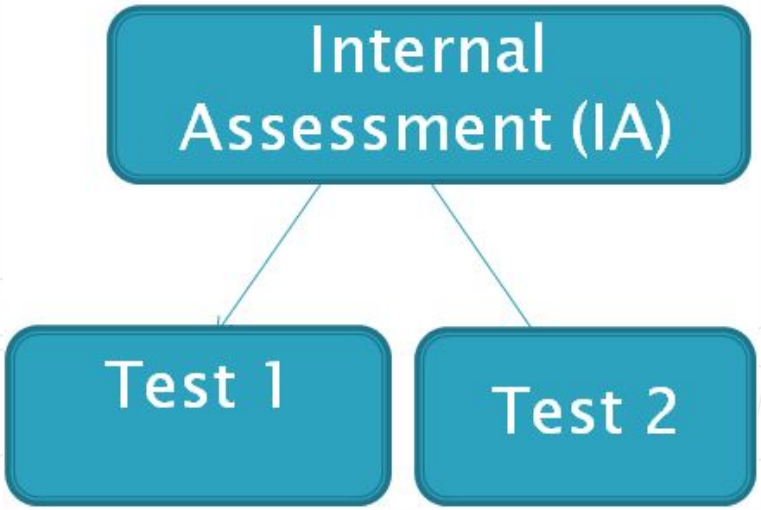
2.2 Protected Members and Constructors in Subclasses

COMPUTER SCIENCE & ENGINEERING

Protected Members

- The private members of a class cannot be directly accessed outside the class. Only methods of that class can access the private members directly.
- In some cases a subclass to access a private member of a superclass.
- A member of a superclass needs to be (directly) accessed in a subclass and yet still prevent its direct access outside the class, can be achieved by **protected member**.

Implementation example



- Series of classes to describe two levels of IA : Test 1 and Test 2. These two levels have certain common properties sub1, sub2, sub3
- This could be represented in the base class IA from which we would derive the two other ones : Test1 and Test2

Protected Members Sample Program

```
//IA.java
public class IA
{
        protected double sub1;
        protected double sub2;
        protected double sub3;
public void setValues(double sub1,double sub2, double sub3)
        {
                this.sub1=sub1;
                this.sub2=sub2;
                this.sub3=sub3;
        }
}
```

```
//Test1.java
public class Test1 extends IA
{
        public double getTotal()
        {
                return(sub1+sub2+sub3);
                //accessing protected members
        }
}
//Test2.java
public class Test2 extends IA
{
        public double getTotal()
        {
                return(sub1+sub2+sub3);
                //accessing protected members
        }
}
```

Protected Members Sample Program

```
public class testprogram
{
        public static void main(String ar[])
        {
                Test1 t1=new Test1();
                t1.setValues(50,60,70);
                System.out.println("Test 1 total  "+t1.getTotal());
                Test2 t2=new Test2();
                t2.setValues(80,85,90);
                System.out.println("Test 2 total  "+t2.getTotal())
        }
}
```

Output

```
D:\cse>javac Test1.java

D:\cse>javac Test2.java

D:\cse>javac testprogram.java

D:\cse>java testprogram
Test 1 total  180.0
Test 2 total  255.0

D:\cse>
```

Constructors in sub classes

- The class which inherits the properties of other is known as subclass and the class whose properties are inherited is known as superclass. **extends** is the keyword used to inherit the properties of a class
- A subclass can have its own private data members, so a subclass can also have its own constructors (is special method that is called when an object is instantiated).
- The constructors of the subclass can initialize only the instance variables of the subclass. Thus, when a subclass object is instantiated the subclass object must also automatically execute one of the constructors of the superclass.
- To call a superclass constructor the **super** keyword is used.

SAMPLE PROGRAM

```
class Super
{
   String s;
   public Super()
      {
         System.out.println("Super");
       }
}
public class Sub extends Super
{
       public Sub()
       {
       System.out.println("Sub");
       }
       public static void main(String[] args)
       {
              Sub s = new Sub();
       }
}
```

OUTPUT OF THE PROGRAM

EXPLANATION

- When inheriting from another class, super() has to be called first in the constructor. If not, the compiler will insert that call. This is why super constructor is also invoked when a Sub object is created.
- This doesn't create two objects, only one Sub object. The reason to have super constructor called is that if super class could have private fields which need to be initialized by its constructor.
- After compiler inserts the super constructor, the sub class constructor looks like the following:

```
public Sub()
    {
super();
System.out.println("Sub");
        }
```

VIDEO LINK

https://www.youtube.com/watch?v=SKBc_A6saCo