Java lecture-7

Unit-II

Inheritance

- Using inheritance, you can create a general class that defines traits common to a set of related items.
  I
- This class can then be inherited by other, more specific classes, each adding those things that are unique to it.

- In the terminology of Java, a class that is inherited is called a *superclass*.

- The class that does the inheriting is called a *subclass*.

- Therefore, a subclass is a specialized version of a superclass.

- It inherits all of the members defined by the superclass and adds its own, unique elements.

- To inherit a class, you simply incorporate the definition of one class into another by using the **extends** keyword.

```
// A simple example of inheritance.
    // Create a superclass.
     class A {

      int i, j;
      void showij() {
         System.out.println("i and j: " + i + " " + j);

} }
    // Create a subclass by extending class A.
     class B extends A {

      int k;
      void showk() {
         System.out.println("k: " + k);

}
    void sum() {
    System.out.println("i+j+k: " + (i+j+k));

} }
```

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

   A superOb = new A();
    B subOb = new B();

   // The superclass may be used by itself.
    superOb.i = 10;
    superOb.j = 20;
    System.out.println("Contents of superOb: ");
    superOb.showij();

   System.out.println();

     /* The subclass has access to all public
members of its superclass. */

    subOb.i = 7;
     subOb.j = 8;
     subOb.k = 9;
     System.out.println("Contents of subOb: ");
     subOb.showij();

    subOb.showk();
     System.out.println();

System.out.println("Sum of i, j and k in subOb:");
 subOb.sum();
  }

}
```

The output from this program is shown here:

```
Contents of superOb:

i and j: 10 20


Contents of subOb:

i and j: 7 8

k: 9


Sum of i, j and k in subOb:

i+j+k: 24
```

- As you can see, the subclass **B** includes all of the members of its superclass, **A**.
- This is why **subOb** can access **i** and **j** and call **showij( )**.
- Also, inside **sum( )**, **i** and **j** can be referred to directly, as if they were part of **B**.
- Even though **A** is a superclass for **B**, it is also a completely independent, stand-alone class.
- Being a superclass for a subclass does not mean that the superclass cannot be used by itself.
- Further, a subclass can be a superclass for another subclass.

- The general form of a **class** declaration that inherits a superclass is shown here:

```
class subclass-name extends superclass-name {
 // body of class
}
```

- You can only specify one superclass for any subclass that you create.

- Java does not support the inheritance of multiple superclasses into a single subclass.

- You can, as stated, create a hierarchy of inheritance in which a subclass becomes a superclass of another subclass.

- However, no class can be a superclass of itself.

Member Access and Inheritance

- Although a subclass includes all of the members of its superclass, it cannot access those members of the superclass that have been declared as **private**.

Using super

- Whenever a subclass needs to refer to its immediate superclass, it can do so by use of the keyword **super**.

- **super** has two general forms.

- The first calls the superclass' constructor.

- The second is used to access a member of the superclass that has been hidden by a member of a subclass.

Using super to Call Superclass Constructors

- A subclass can call a constructor defined by its superclass by use of the following form of **super**:

  super(*arg-list*);

  Here, *arg-list* specifies any arguments needed by the constructor in the superclass.

- **super( )** must always be the first statement executed inside a subclass' constructor.

```
// BoxWeight now uses super to initialize its Box attributes.


    class BoxWeight extends Box {
     double weight;         // weight of box
     // initialize width, height, and depth using super()
      BoxWeight(double w, double h, double d, double m) {
       super(w, h, d);          // call superclass constructor
       weight = m; }
}
```

A Second Use for super

The second form of **super** acts somewhat like **this**, except that it always refers to the superclass of the subclass in which it is used.

This usage has the following general form:

super.*member*

Here, *member* can be either a method or an instance variable.

This second form of **super** is most applicable to situations in which member names of a subclass hide members by the same name in the superclass.