CS205 Object Oriented Programming in Java

Module 2 - **Core Java Fundamentals (Part 9)**

Prepared by

Renetha J.B.

AP

Dept.of CSE,

Lourdes Matha College of Science and Technology

Topics

- Core Java Fundamentals**:**
  - ✓Inheritance :
    - ✓ Super class
    - ✓Sub class
    - ✓keywords ***super***
    - ✓ ***protected*** Members

Inheritance

- Inheritance helps to create hierarchical classifications.
- Using inheritance we can create a **general class(base or super class)** that defines features c**ommon** to a set of related items.
  - This class can then be **inherited** by other, more **specific classes(subclasses).**

Inheritance(contd.)

- A subclass is a specialized version of a superclass.
- Subclass **inherits all of the instance variables and methods defined** by the superclass and adds its own, unique elements.
- To inherit a class, we have to use **extends** keyword along *with subclass definition.*

class superclass**{ //statements……}**

class subclass **extends** superclass**{ //statements……}**

```
// A simple example of inheritance.
class A
{
    int i, j;
    void showij()
    {
    System.out.println("i and j: " + i + " " + j);
    }
}
class B extends A {
int k;
void showk() {
                    System.out.println("k: " + k);
                    }
void sum() {
          System.out.println("i+j+k: " + (i+j+k));
                }
}
```

Here A is the superclass of B

```
class A
{
    int i, j;
void showij()
{
System.out.println(i + "  " + j);
}
}
class B extends A
 {
int k;
void showk() {
   System.out.println("k: " + k);
                    }
void sum() {
System.out.println("sum " + (i+j+k));
                }
}
```

```
class SimpleInheritance
{
public static void main(String args[]) {
A superOb = new A();
B subOb = new B();
superOb.i = 10;
superOb.j = 20;
System.out.println("Superobj Contents ");
superOb.showij();
subOb.i = 7;
subOb.j = 8;
subOb.k = 9;
System.out.println("subOb contents ");
subOb.showij();
subOb.showk();
System.out.println("Sum in subOb:");
subOb.sum(); } }
```

```
Superobj Contents
10 20
subOb contents
7 8
k: 9
Sum in subOb:24
```

Member Access and Inheritance

- Subclass cannot access the private members in superclass.

```
class A {
int i;    // public by default
private int j; // private to A
void setj(int x) { j = x; };
}
class B extends A {
int total;
void sum()     {
total = i + j; // ERROR, j(private) is not accessible here
                    }
}
```

A class member that has been declared as private will remain private to its class.
It is not accessible by any code outside its class, including subclasses.

- A major advantage of inheritance is that **once you have created a superclass** that defines the attributes **common** to a set of objects, it can be used to create any number of more specific subclasses.
- Each subclass can have its own special features also.

A Superclass Variable Can Reference a Subclass Object

- A **reference variable of a superclass** can be assigned a reference to any subclass derived from that superclass.

```
class A
{
}
class B extends A
{
}
class Sample
{A oba=new A();
B obb=new B();
oba=obb; }
```

Superclassobject=subclassobject

When a reference to a subclass object is assigned to a superclass reference variable, we will have **access only to those parts of the object defined by the superclass**..

```
class Sup
{
int a,b;
void area()
{
System.out.println("Product="+ a*b);
}

}
class Sub extends Sup
{
int i;
Sub(int x,int y,int z)
{
a=x;
b=y;
i=z;
}

}
```

```
class InhRefsub{

public static void main(String args[])
  {
  Sup supob=new Sup();
  supob.area();
  Sub subob=new Sub(10,20,30);
  supob=subob;
  supob.area();
//System.out.println("i="+ supob.i);//ERROR
  }
}
```

OUTPUT

Product=0

Product=200

Program explanaion

- Here the statement **Sup supob=new Sup();** creates an object of class Sup named supob using default constructor **Sup().** Supob has variables a and b. Since default constructor is not there, compiler provides default constructor by initializing all variables to zero, so a and b are initially 0.
- Next **supob.area();** will call area() in Sup and prints *Product=0*
- **Sub subob=new Sub(10,20,30);** creates object of Sub named **subob** using parameterized constructed ***Sub(int x,int y,int z)*** . Since Sup is the subclass of Sub, so Sub has variables a,b from Sup and i (own variable) and set a=10 b=20 i=30
- The statement **supob=subob;** assigns object **subob** to superclass object reference **supob.** So supob has value of a and b(superclass variables) same as subob. a=10 b=20
  supob.area(); will print *Product=200*

Using super

- Whenever a subclass needs to refer to its immediate superclass, it can be done using the keyword **super.**
- **super** has two general forms**.**
  1. To call **the superclass' constructor.**
  2. To **access a member of the superclass** that has been hidden by a member of a subclass.

❑ The **static methods** cannot refer to **super.**

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.

```
class Sup
{
Sup()
{
System.out.println("Superclass");
}
}
class Sub extends Sup
{

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

}
```

```
class Supersub{

public static void main(String args[])
  {

  Sub subob=new Sub();

  }
}
```

OUTPUT
Superclass
Subclass

super keyword to acess member

- **super** always refers to the superclass of the subclass in which it is used.
- To access the member in superclass from subclass

  super.*member*
  - *Here member can be either a method or an instance variable.*
- If subclass contains same variable as superclass, then in subclass, the superclass member will be hidden by corresponding subclass member.
  - This can be prevented using super keyword

```
class A    {
int i;
            }

class B extends A
{
int i;                  // this i hides the i in A

B(int a, int b)         {
super.i = a;            // i in A
i = b;                  // i in B
                        }

void show()             {
System.out.println("i in superclass: " + super.i);
System.out.println("i in subclass: " + i);
                        }
}
```

```
class UseSuper {
public static void main(String args[])
    {
B subOb = new B(1, 2);
subOb.show();
}
}
```

OUTPUT
i in superclass: 1
i in subclass: 2

Creating multiple hierarchy


```
class A
{
int x;
A(int p)
{
System.out.println("Superclass A ");
x=p;
}
}
class B extends A
{
int y;
B(int p,int q)
{
super(p);
System.out.println("B Subclass of A");
y=q;
}

}
```

```
class C extends B
{
int z;
C(int p,int q,int r)
{
super(p,q);
System.out.println("C Subclass of A");
z=r;
}

}


class Mulinh{

public static void main(String args[])
    {

   C ob=new C(10,20,30);
     System.out.println("x="+ob.x);
     System.out.println("y="+ob.y);
     System.out.println("z="+ob.x);
          }  }
```

```
Superclass A
B Subclass of A
C Subclass of A
x=10
y=20
z=10
```

Protected members

- Protected members are declared by prefixing the access specifierc protected.

  protected datatype member;

- The protected member in a class can be accessed by
  - any class within the **same** package.
  - direct **sub-classes in other package** also.

Protected members(contd;)

- If you want to allow an element(member) to be **seen** outside your current package, but only to classes that subclass your class directly, then declare that element (member) **protected.**

- **Eg.**

```
class  A
{
protected int c;                    //protected variable
int a;
private char b;
public float f;
protected void add()                //protected method
{  //statements
}
//methods and statements
}
```

Reference

- Herbert Schildt, Java: The Complete Reference, 8/e, Tata McGraw Hill, 2011.