Chapter 4 : Inheritance And Interferance

Inheritance in Java

- Inheritance is an important pillar of OOP(Object-Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class.
- **Important terminology :**
- **Super Class:** The class whose features are inherited is known as superclass(or a base class or a parent class).
- **Sub Class:** The class that inherits the other class is known as a subclass(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.

- **Reusability:** Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

How to use inheritance in Java

The keyword used for inheritance is **extends.**

Syntax :

class derived-class extends base-class { //methods and fields }

There are five types of inheritance single, multilevel, multiple, hybrid and hierarchical.

1. Single level :

In this one class i.e., derived class inherits properties from its parental class. This enables code reusability and also adds new features to the code. Example: class b inherits properties from class a

Class A is base or parental class and class b is derived class

Syntax :

```
Class a {
…
}
Class b extends class a {
…
}
```

2. **Multilevel** :

This one class is derived from another class which is also derived from another class i.e., this class has more than one parental class, hence it is called multilevel inheritance.

Syntax :

```
Class a {
….
}
Class b extends class a {
….
}
Class c extends class b {
…
}
```

3. **Hierarchical level** :

In this one parental class has two or more derived classes or we can say that two or more child classes has one parental class.

Syntax :

```
Class a {
…
}
Class b extends class a {
..
}
Class c extends class a {
..
}
```

4. Hybrid inheritance :

This is the combination of multiple and multilevel inheritance and in java multiple inheritance is not supported as it leads to ambiguity and this type of inheritance can only be achieved through interfaces.

Consider that class a is the parental or base class of class b and class c and in turn class b and class c are parental or base class of class d. Class b and class c are derived classes from class a and class d is derived class from class b and class c.

Following program creates a super class called add and a subclass called sub, uses extend keyword to create a subclass add.

```
//create a superclass
Class Add
{
int my;
int by;
void setmyby (int xy, int hy) {
my=xy;
by=hy;
}
}
```

```
//create a sub class
class b extends add {
int total;
void sum () {
public Static void main (String args [ ] ) {
b subOb= new b ( );
subOb. Setmyby (10, 12);
subOb. Sum ( ) ;
System.out.println("total =" + subOb. Total);
}
}
```

Output :

total = 22

Access in subclass

The following members can be accessed in a subclass:

1)public or protected superclass members.

2)Members with no specifier if subclass is in same package.

The “super” keyword

It is used for three purposes:

Invoking superclass constructor - super(arguments)
Accessing superclass members – super.member
Invoking superclass methods – super.method(arguments)

Example :

```
class A
{ protected int num;
  A(int num) { this.num = num; }
  class B extends A
{
   int num;
   B(int a, int b) {
     super(a); //should be the first line in the subclass constructor
     this.num = b;
    }
      void display() { System.out.println("In A, num = " +
      super.num); System.out.println("In B, num = "+ num);
    }
}
```

Overriding methods

Redefining superclass methods in a subclass is called overriding. The **signature** (structure of method i.e. name, return type and parameters) of the subclass method should be the same as the superclass method.

```
class A
{
 void method1(int num) {
 }//code
 }
```

```
class B extends A
{
 void method1(int x) {
  //code
 }
}
```

Dynamic binding

When over-riding is used, the method call is resolved during run-time i.e. depending on the object type, the corresponding method will be invoked.

Example *:*

```
A ref;
ref = new A();
ref.method1(10); //calls method of class A
ref = new B();
ref.method1(20); //calls method of class B
```

Abstract class :

An abstract class is a class which cannot be **instantiated**(can con create an object of the class). It is only used to create subclasses. A class which has abstract methods must be declared abstract. An abstract class can have data members, constructors, method definitions and method declarations.

```
abstract class ClassName
{
  ...
}
```

Abstract method :

An abstract method is a method which has no definition. The definition is provided by the subclass.

```
abstract returnType method(arguments);
```

Interface

An interface is a pure abstract class i.e. it has only abstract methods and final variables. An interface can be implemented by multiple classes.

```
interface InterfaceName
{
 //abstract methods
 //final variables
}
```

Example :

```
interface MyInterface
{
```

```
  void method1(); void method2();
  int size= 10; //final and static
}
class MyClass implements MyInterface {
  //define method1 and method2
}
```

Sample program to demonstrate inheritance and interfaces

```
interface Shape
{
double area();
}
class Circle implements Shape
{
```

```
double radius; Circle(double radius)      {
this.radius=radius;
}
public double area()          {
return java.util.Math.PI * radius* radius;
}
}
class Cylinder extends Circle
{
double height;
Cylinder(double radius, double height)  { super(radius);
this.height=height;
}
public double area()          //overriding
```

```
{
return java.util.Math.PI * radius* radius *height;
}
}
public class Test {
public static void main(String[] args)
{
Shape s;
s = new Circle(5.2);
System.out.println("Area of circle = " + s.area()); s = new Cylinder(5,
2.5);
System.out.println("Area of cylinder = " + s.area());
}
```