CS F213
Object Oriented Programming

BITS Pilani
Hyderabad Campus

Prof.R.Gururaj
CS&IS Dept.

Inheritance

Ch.8 of T1.

The Complete Reference- Java, 11th Edition, Herbert Schildt, Tata McGraw Hill Publishing.

And also refer to Class notes.

Content covered

innovate achieve lead

1. Inheritance basics
2. Member access and inheritance
3. Superclass referencing subclass object and use of *super* keyword
4. Multilevel hierarchy
5. Constructor calling
6. Method overriding
7. Dynamic method dispatch
8. Abstract classes
9. Using *final* with inheritance
10. The *Object* class

Inheritance

This allows us to create hierarchical classifications.

One class can inherit traits/properties of another class.

class that is inherited is called *superclass*.

Hence, *subclass* is specialized version of superclass.

We use the keyword *extends* for inheriting a class from other

Member access

- If super class member is not accessible to a subclass if it is specified as *private*

```
class A
{    private int a;
     void showSuper()
     {System.out.println(" Value of a is ::"+a);}
}
class B extends A
{    int b;
      void showSub()
      {System.out.println(" Value of a and b are ::"+a+" and "+b);}
}
```

This will not compile because class B trying to access a private member 'a' of A

-A reference variable of superclass can be used to point to any subclass object of that superclass.

-In such case we can access only those parts of subclass objects which defined in super class.

Valid

A aObj; aObj=new B();

aObj.a=29; aObj.test();

invalid

A aObj; aObj=new B();

aObj.c=90; aObj.add();

‘super’ keyword

The keyword *super is used in two ways*

1. *super() - Call super class constructor from subclass constructor, and must be the first statement in sub class constructor*
2. *Access super class members when subclass members hide superclass members.*

It is possible to declare a variable in subclass with same name as declared in superclass.

In such case subclass members hide superclass members.

```
class SuperDemo
{
    public static void main(String org[])
    {
       BoxWeight bw=new BoxWeight(2,3,4,5);
     }
}
```

Creating multilevel hierarchy

```
class A
{
    int a;
}
class B extends A
{
    int b;
}
class C extends B
{
    int c;
}
```

super() in class B calls A()

super() in class C calls B()

If super() is not used in subclass then default or parameterless constructor of superclass is used.

```
class A
{
    A(){System.out.println(" Inside A's constructor :");
}
class B extends A
{
    B(){System.out.println(" Inside B's constructor :");
}
class C extends B
{
    C(){System.out.println(" Inside C's constructor :");
}
```

```
class ConstructorsDemo
{
    public static void main(String org[])
    {
        C cobj=new C();
    }
}
```

Output:

Inside A’s constructor :

Inside B’s constructor :

Inside C’s constructor :

Method Overriding

- subclass can have a method defined, which has same name and type signature (arguments & return type) as the one available in its super class.
- Then the subclass is said to override the method in its superclass.
- When a overridden method is called from the subclass it always refers to the version defined in the sub class.
- Why overriding: superclass can specify methods that will be common to all its derivatives this implements ‘one interface multiple methods’ aspect of polymorphism.

Dynamic Method Dispatch

- Is a mechanism by which a call to an overridden method is resolved at runtime, rather than compile time.
- This is how Java implements run-time polymorphism.

```
class OverrideDemo
{
        public static void main(String org[])
        {
        A a=new A();
        a.test(3); this will invoke A's test()
        a=new B();
        a.test(7); this will invoke B's test()
         }
}
```

Dynamic Method Dispatch P

Abstract class

- Superclass declares the structure of the class, without complete implementation to all the methods(abstract class).
- Here superclass is unable to define the implementation.
- It is the responsibility of the sub class to provide implementation required.
- As there is no completeness an abstract class object can not be instantiated.
- But we can declare a reference variable of an abstract class and make it to point to some concrete subclass object.

Use of final keyword

The final keyword for methods A super class method specified with final modifier can not be overridden by any of its subclasses.

Ex. *final void add(int a, int b){ // body }*

The final keyword for class A class specified with final modifier can not be inherited.

Ex. *final class Box { //body }*

We can declare a variable as final meaning that it can't be modified.

Ex. *final int VAL=9;*

The Object class - is the super class for all java classes

Object clone()

void finalize()

boolean equals(Object o)

void notify() //final

void notifyAll() //final

void wait() // final

String toString()

Object typecasting

```
class TypeCastDemo1
{
    public static void main(String org[])
    {
        Line l;
         l=new StLine(); meth(l);
    }
    static void meth(Line l)
    {
        StLine l1=l; l1.print1();
        Line ln=l; ln.print();
    }
}
```

```
C:\Users\Admin\JavaPrograms>javac TypeCastDemo1.java
TypeCastDemo1.java:25: error: incompatible types: Line cannot be
    converted to StLine
            StLine  l1=l;
                    ^
1 error
```

```
class TypeCastDemo1
{
    public static void main(String org[])
    {
        Line l;
         l=new StLine(); meth(l);
    }
    static void meth(Line l)
    {
        StLine l1=(StLine)l;
         l1.print1();
        Line ln=l; ln.print();
    }
}
```

Summary

1. Inheritance basics
2. Member access and inheritance
3. Superclass referencing subclass object and use of *super* keyword
4. Multilevel hierarchy
5. Constructor calling
6. Method overriding
7. Dynamic method dispatch
8. Abstract classes
9. Using *final* with inheritance
10. The *Object* class
11. Object type Casting