BITS Pilani

Object-Oriented Programming (CS F213)

Module III: Inheritance and Polymorphism in Java

CS F213 RL 9.2: Inheritance Basics-II

Dr. Pankaj Vyas
Department of Computer Science, BITS-Pilani, Pilani Campus

CS F213 RL 9.2 : Topics

- Inheritance Basics - II

Inheritance Basics

- Suppose A is super class of B. Then class B can inherit all the features (Methods and Instance Fields) of class A except the private members.
- Super and Sub Classes May Either Belong to Same or Different Packages

Access Modifiers

Access Locations

	public	protected	package- private	private
With in the Same Class	Yes	Yes	Yes	Yes
Sub-Classes in same package	Yes	Yes	Yes	No
Other Classes in same package	Yes	Yes	Yes	No
Subclasses in other packages	Yes	Yes	No	No
Non-subclasses in other packages	Yes	No	No	No

Inheritance Basics ….

- Whenever an instance of sub-class type is created, a super-class constructor is called first.
- If a super-class constructor does not have any constructor of its own OR has an un-parametrized constructor then it is automatically called by JRE by using call super()
- If a super-class has a parameterized constructor then it is the responsibility of the sub-class constructor to call the super class constructor by call

super(<parameters required by super class>)

- Call to super class constructor must be the first statement in sub class constructor

Extending Classes : Example 1

innovate achieve lead

When super-class has an Un-parametrized Constructor

```
// File Name : Sample.java
class A
{
		A()
		{
				System.out.println("This is constructor of class A");
		}
} // End of class A
class B extends A
{
		B()
		{
				super();
				System.out.println("This is constructor of class B");
		}
} // End of class B
class Test
{
		public static void main(String[] args)
		{
				B		b1		=		new		B();
		}// End of Method
}// End of Class Test
```

Call to a Constructor Method of super-class

OUTPUT

```
This is constructor of class A
This is constructor of class B
```

Extending Classes : Example 2

When a super-class has Either no or one Un-parametrized Constructor

```
// File Name : Sample.java
class A
{
        A()
        {
                System.out.println("This is constructor of class A");
        }
} // End of class A
class B extends A
{
        B()
        {
                System.out.println("This is constructor of class B");
        }
} // End of class B
class Test
{
        public static void main(String[] args)
        {
                B        b1        =        new        B();
        }// End of Method
}// End of Class Test
```

No super() statement

OUTPUT

```
This is constructor of class A
This is constructor of class B
```

Extending Classes : Example 3

When super-class has only a Parametrized Constructor

```
// File Name : Sample.java
class A
{
            A(int x, int y)
            {
                        System.out.println("This is constructor of class A");
            }
} // End of class A
class B extends A
{
            B()
            {
                        System.out.println("This is constructor of class B");
            }
} // End of class B
class Test
{
            public static void main(String[] args)
            {
                        B          b1          =          new          B();
            }// End of Method
}// End of Class Test
```

Results in Compile-Time Error.

A sub-class constructor does not class super-class constructor

Extending Classes : Example 3

When super-class has only a Parametrized Constructor

```
// File Name : Sample.java
class A
{
            A(int x, int y)
            {
                        System.out.println("This is constructor of class A");
            }
} // End of class A
class B extends A
{
            B(int a, int b)
            {
                        super(a,b);

                        System.out.println("This is constructor of class B");
            }
} // End of class B
class Test
{
            public static void main(String[] args)
            {
                        B          b1          =          new          B();
            }// End of Method
}// End of Class Test
```

A sub-class constructor must call a parameterized constructor of super-class

Note : super() statement must be the first statement in a sub-class constructor

Extending Classes : Example 4

‘final’ classes cannot have sub-classes

```
// File Name : Sample.java
final class A
{
		A(int x, int y)
		{
				System.out.println("This is constructor of class A");
		}
} // End of class A
class B extends A
{
		B(int a, int b)
		{
				super(a,b);

				System.out.println("This is constructor of class B");
		}
} // End of class B
```

The Code Will Result in Compile-Time Error.

Role of super Keyword

- 'super' Java keyword primarily used for two purposes

1. To Call constructor method of super-class from a sub-class constructor

   Syntax:

   (i) super() ; // if super-class has only one un-parameterized constructor

   (ii) super(<parameters>); // if super-class has only one parameterized constructor

2. To call a method of super-class in a sub-class method or to access an instance field of super-class in sub-class especially when their names are similar

   Syntax:

   (i) super.<super-class-method()> ;

   (ii) super.<super-class-instance-field>;

Role of super : Example 1

```
// File Name : XYZ.java
class A
{
    private int a;                  // instance-field
    // Constructor Method
    A( int a)
    {
       this.a =a;
      System.out.println("This is constructor of class A");
    } // End of Constructor
    // print Method
    void print()
    {
       System.out.println("a="+a);
    } // End of Method
   // Display Method
    void display()
    {
       System.out.println("hello This is Display in A");
    } // End of Method
} // End of class A
```

```
class B extends A
{
   private int b;          // instance field
   private double c;   // instance field
   B(int a,int b,double c)
   {
       super(a);
       this.b=b;
       this.c=c;
       System.out.println("This is constructor of class B");
   }// End of Constructor Method
   // show  Method
   void show()
   {
       print();
       System.out.println("b = " + b);
       System.out.println("c = " + c);
    }// End of Method
} // End of class B
```

Call to print() method of super-class
Can be Written as super.print()

Call to super-class
Constructor

Role of super : Example 2

```
// File Name : XYZ.java
class A
{
        protected           int     a = 20;         // instance-field
} // End of class A
class B extends A
{
        private             int     a = 30;
        void show()
        {
                int     a = 50;
                System.out.println(“ a= “ + a);
                System.out.prinltn(“ a= “+ this.a);
                System.out.println(“ a = “+ super.a);
        }// End of Method
}// End of class B
```

Role of super : Example 3

```
// File Name : XYZ.java
class A
{
    private int a;                // instance-field
    // Constructor Method
    A( int a)
    {
       this.a =a;
      System.out.println("This is constructor of class A");
    } // End of Constructor
    // print Method
    void print()
    {
       System.out.println("a="+a);
    }
   // Display Method
    void show()
     {
        System.out.println("Hello This is Display in A");
     }
} // End of class A
```

```
class B extends A
{
   private int b;         // instance field
   private double c;   // instance field
   // Constructor Method
   B(int a, int b, double c)
   {
      super(a);
      this.b=b;
      this.c=c;
      System.out.println("This is constructor of class B");
   }// End of Constructor Method
   // show  Method
   void show()
    {
     super.show();
     System.out.println("b = " + b);
     System.out.println("c = " + c);
    }// End of Method
} // End of class B
```

Call to show() method of super-class
Can be Written as super.print()

Call to super class Constructor

innovate achieve lead

Thank You