Object Oriented Analysis and Design with Java

UE19CS353

Dr. L. Kamatchi Priya
Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

UE19CS353: Object Oriented Analysis and Design with Java

Inheritance -II

Dr. L. Kamatchi Priya
Department of Computer Science and Engineering

Constructor Calling In Inherited Classes

- Constructors are called in the order of derivation, from superclass to subclass
- super() is the first statement executed in the subclass constructor, irrespective whether super() is explicitly specified or not.
- If super() not specified then default constructor (parameter less) of the super class is called.
- Super class constructor must be called by the subclass constructor first to do the initialization of super class fields/data. (**mandate**)

Constructor Calling In Inherited Classes

```
class Constructor{
      public static void main(String[] args){
      A o1 = new A(); // Constructor A called
      B o2 = new B(); // Constructor A called Constructor B called
      A o3 = new B(); // Constructor A called Constructor B called
      }
}
class A{
      public A(){
      System.out.print("Constructor A called ");
      }
}
class B extends A{
      public B(){
      System.out.print("Constructor B called");
      }
}
```

Method Overriding

- If the method in the subclass has same name and type signature as superclass, then the method in the subclass is said to override the method in the superclass.

```
class Inheritance{
      public static void main(String[]args){
            B obj = new B();
            obj.method(); // method of B
            }
}
class A{
      void method(){
            System.out.println("method of A");
      }
}

class B extends A{
      void method(){
            System.out.println("method of B");
      }
}
```

Method Overriding

- To invoke super class version of the overridden method in sub class, super is used.

```
class Inheritance{
        public static void main(String[]args){
                B obj = new B();
                obj.method();
        }
}
class A{
        void method(){
                System.out.println("method of A");
        }
}

class B extends A{
        void method(){
                super.method();
                System.out.println("method of B");
        }
}
```

Output:

```
method of A
method of B
```

Dynamic Method Dispatch

- Dynamic Method dispatch is a mechanism by which call to an overridden method is resolved at **run-time**.

HOW?

- **Superclass reference variable can refer to subclass objects**
- When an overridden method is called through a superclass reference, Java determines which version of that method to execute based upon the type of the object being referred to at the time the call occurs.

Dynamic Method Dispatch

```
class A {
void callme() {
      System.out.println("Inside A's callme method"); } }
class B extends A {
      void callme() { System.out.println("Inside B's callme method"); } }
class C extends A {
      void callme() { System.out.println("Inside C's callme method"); }}
class Dispatch {
      public static void main(String args[]) {
            A a = new A();
            B b = new B();
            C c = new C();
            A r;
            r = a; r.callme(); // calls A's version of callme
            r = b; r.callme(); // calls B's version of callme
            r = c; r.callme(); // calls C's version of callme } }
```

THANK YOU

Dr. L. Kamatchi Priya

Department of Computer Science and Engineering

priyal@pes.edu