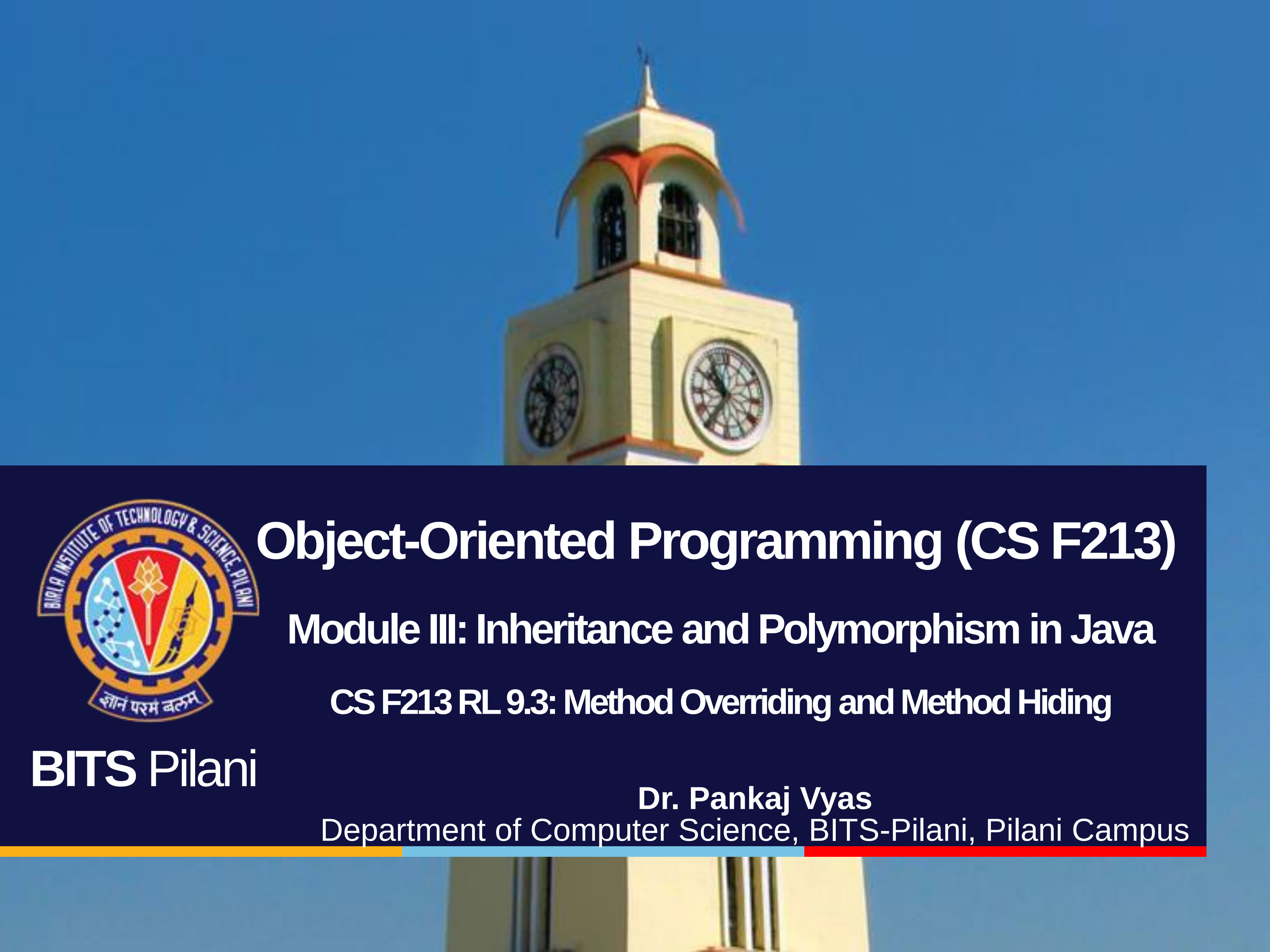


Object-Oriented Programming (CS F213)

Module III: Inheritance and Polymorphism in Java

CS F213 RL 9.3: Method Overriding and Method Hiding

BITS Pilani

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

CS F213 RL 9.3 : Topics

- Method Overriding in Java
- Method Hiding in Java

Method Overriding in Java

- Also known as Runtime Polymorphism, Dynamic Method Dispatch or Late Binding
- A sub-class overrides an object (instance) method of a super-class if the following three conditions are satisfied
  1. The method of a sub-class and method of a super class have same name
  2. The method of a sub-class and method of a super class have same signature
  3. The method of a sub-class and method of a super class have same return type
- The scope of a sub-class method should be either same or higher than that of a super-class method
- Note : private methods of a super-class are not visible in sub-class. Hence, a sub-class cannot override the private methods of super class.
- Call to an overridden method is decided at runtime.
- Call to a overridden method is not decided by the type of reference variable. Rather it is decided by the type of the object where reference variable is pointing.

Method Overriding : Example

```
class A
{
        void show(int a, int b)
        {
                System.out.println("Hello ! This is show() in A");
        }// End of show() Method

} // End of class A
class B extends A
{
        void show(int a, int b)
        {
                System.out.println("Hello This is show() in B");
        }// End of show() Method

} // End of class B
// Driver Class
class       Test
{
        public      static      void      main(String args[])
        {

                A       a1      =       new     A();
                a1.show(4,5);

                a1      =       new     B();
               a1.show(1,1);
        }// End of Method
}// End of class Test
```

sub-class B class overrides the show() Method of super-class A

Calls show() from B class

Calls show() from A class

Type of Variable ‘a1’ is A
Type of Object is ‘A’

Type of Variable ‘a1’ is still A
But type of Object is ‘B’

Basic Facts About Method Overriding : Fact I

- A super class variable can point to any of its sub-class instance
- Example : Suppose 'X' is the super-class for sub-classes 'A', 'B' and 'C'.

```
X        x1       =        new       A();
```

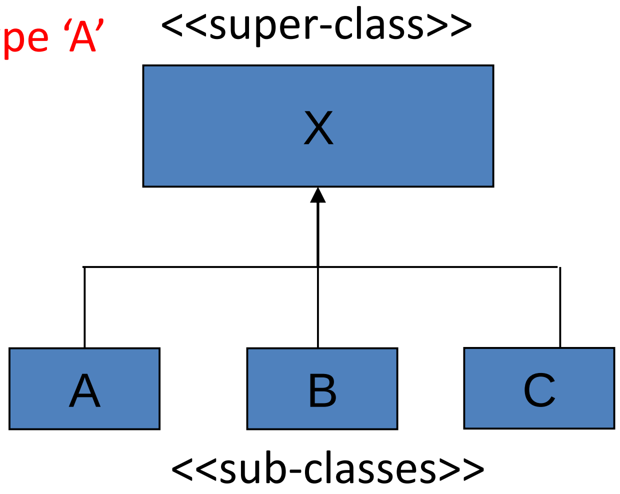
A super-class variable of type 'X'
pointing to a sub-class instance of type 'A'

```
x1       =        new       B();
```

A super-class variable of type 'X'
pointing to a sub-class instance of type 'B'

```
x1       =        new       C();
```

A super-class variable of type 'X'
pointing to a sub-class instance of type 'C'



Basic Facts About Method Overriding : Fact II

- When a super-class variable points to any of its sub-class instance, then from the sub-class only overridden methods can be invoked by using the same variable

```
class A
{
        void show()
        {
        }// End of Method
}// End of class A

class B extends A
{
        void show()
        {
        } / End of Method

        void display()
        {
        } / End of Method

}// End of class B
```

B class overrides show() Method

A a1 = new B();

a1.show(); $\Rightarrow$ **Valid Statement**

a1.display(); $\Rightarrow$ **Results in Compile-Time Error**

Basic Facts About Method Overriding : Fact III

- Overridden Method in sub-class should have either same or higher scope than the scope of a similar method in the super class.

```
class A
{
        void show()
        {
        }// End of Method
}// End of class A

class B extends A
{
        public void show()
        {
        } / End of Method

        void display()
        {
        } / End of Method

}// End of class B
```

A's show() method has package-private scope

B's show() method has public scope

NO ERROR

Basic Facts About Method Overriding : Fact III …

- Overridden Method in sub-class should have either same or higher scope than the scope of a similar method in the super class. The scope of overridden method in sub-class should not have lesser scope

```
class A
{
        protected void show()
        {
        }// End of Method
}// End of class A

class B extends A
{
        public void show()
        {
        } / End of Method

        void display()
        {
        } / End of Method

}// End of class B
```

A's show() method has protected scope

B's show() method has public scope

NO ERROR

Basic Facts About Method Overriding : Fact III …

- Overridden Method in sub-class should have either same or higher scope than the scope of a similar method in the super class.

```
class A
{
        protected void show()
        {
        }// End of Method
}// End of class A

class B extends A
{
        void show()
        {
        } / End of Method

        void display()
        {
        } / End of Method

}// End of class B
```

A's show() method has protected scope

B's show() method has package-private scope

COMPILE-TIME ERROR

Basic Facts About Method Overriding : Fact IV

- 'final' Methods of super-class cannot be overridden by sub-class.

```
class A
{
        final void show()
        {
        }// End of Method
}// End of class A

class B extends A
{
        void show()
        {
        } / End of Method

        void display()
        {
        } / End of Method

}// End of class B
```

A's show() method is final

B cannot override final method from super class A

COMPILE-TIME ERROR

Find the Error ?

```
class A
{
        int show(int a, int b)
        {
        }// End of Method

}// End of class A

class B extends A
{
        void show(int a, int b)
        {
        } / End of Method

        void display()
        {
        } / End of Method

}// End of class B
```

Compile-Time Error
(Overridden Methods Cannot Have Different Return Types)

Find the Error ?

```
class A
{
        void show(int a, int b)
        {
        }// End of Method

}// End of class A

class B extends A
{
        void show()
        {
        } / End of Method

        void display()
        {
        } / End of Method

}// End of class B
```

No Error

This Code Represents Method Overloading

```
B      b1      =      new      B();

b1.show();

b1.show(10,8);
```

Calls show() from A class

Calls show() from B class

Does the Following Code Snippet Represent Method Overriding?

```
class A
{
                    private void show(int a, int b)
                    {
                       System.out.println("A : " + (a+b));
                    }// End of Method

}// End of class A
class B extends A
{
                    void show(int a, int b)
                    {
                         System.out.println("A : " + (a*b));
                    } / End of Method
                    void display()
                    {
                    } / End of Method
}// End of class B
```

Method is Private
So, Not Visible in class B

NO. IT DOES NOT REPRESENT METHOD OVERRIDING

Method Hiding

- A sub-class can hide a static method (class method) of a super-class if the following conditions are satisfied for class methods
  1. Methods should have same name, same signatures and same return type
  2. Methods should be static and not instance (or object)

Method Hiding: Example 1

```
class A
{
        public static void show(int a, int b)
        {
                System.out.println(" Class A Method :" + (a + b));
        }// End of Method
}// End of class A
class B extends A
{
        public static void show(int a, int b)
        {
                System.out.println(" Class B Method :" + (a * b));
        } // End of Method
}// End of class B
class Test
{
        public  static  void   main(String[] args)
        {
                A          a1          =          new          A();
                a1.show(5,6);

                a1          =          new          B();
                a1.show(5,6);
        }// End of Method
}// End of class Test
```

class B hides a static method of super-class A

OUTPUT

```
class A Method: 11
class A Method: 11
```

Method Hiding: Example 2

```
class A
{
        protected static void show(int a, int b)
        {
                System.out.println(" Class A Method :" + (a + b));
        }// End of Method
}// End of class A
class B extends A
{
        public static void show(int a, int b)
        {
                System.out.println(" Class B Method :" + (a * b));
        } // End of Method
}// End of class B
class Test
{
        public  static  void   main(String[] args)
        {
                A          a1          =          new          A();
                a1.show(5,6);

                a1          =          new          B();
                a1.show(5,6);
        }// End of Method
}// End of class Test
```

class B hides a static method of super-class A

OUTPUT

```
class A Method: 11
class A Method: 11
```

Method Hiding: Example 3

```
class A
{
        public static void show(int a, int b)
        {
                System.out.println(" Class A Method :" + (a + b));
        }// End of Method

}// End of class A
class B extends A
{
        public void show(int a, int b)
        {
                System.out.println(" Class B Method :" + (a * b));
        } // End of Method

}// End of class B
class Test
{
        public  static  void   main(String[] args)
        {
                A           a1           =           new           A();
                a1.show(5,6);

                a1           =           new           B();
                a1.show(5,6);
        }// End of Method
}// End of class Test
```

COMPILE-TIME ERROR

Method Hiding: Example 3

```
class A
{
        public static void show(int a, int b)
        {
                System.out.println(" Class A Method :" + (a + b));
        }// End of Method

}// End of class A
class B extends A
{
        protected static void show(int a, int b)
        {
                System.out.println(" Class B Method :" + (a * b));
        } // End of Method

}// End of class B
class Test
{
        public  static  void   main(String[] args)
        {
                A         a1         =         new         A();
                a1.show(5,6);

                a1        =          new        B();
                a1.show(5,6);
        }// End of Method
}// End of class Test
```

COMPILE-TIME ERROR

Thank You