Object-Oriented Programming (CS F213)

Module III: Inheritance and Polymorphism in Java

CS F213 RL 10.4: Nested and Inner Classes

BITS Pilani

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

CS F213 RL 10.3 : Topics

- Nested Classes

Nested Class

- Class With-in The Boundary of Another Class
- Two Forms : 1. Static-Nested Class
  2. Non-Static Nested Class

```
class Outer
{
    static  class   static_nested
    {
    }// End of class nested_static

    class   non_static_nested
    {
    }// Endof class non_static_nested
}// End of class Outer
```

Static Nested Class

Non-Static Nested Class

Static Nested Class

- Static keyword applied for class declaration
- Static Nested class can use the instance-fields/object-methods of the outer class only through object reference.
- Static Nested class can be Accessed outside the outer class [provided its scope is not private] using the following syntax

  <Outer-Class-Name>.<Static-Nested-Class-Name>
- *To create an object for the static-nested-class, use this syntax:*

```
<Outer-Class-Name>.<Static-Nested-Class-Name>  nested-class-instance-name
                                =
        new  <Outer-Class-Name>.<Static-Nested-Class-Name>();
```

OR

```
<Outer-Class-Name>.<Static-Nested-Class-Name>  nested-class-instance-name
                                =
        new  <Outer-Class-Name>.<Static-Nested-Class-Name>(<parameters>);
```

Static-Nested Class : Example 1

```
class A
{
        private int a;
        A(int a)
        {
        this.a =a;
        }
        void print()
        {
                System.out.println("a="+a);
        }
        static      void       display()
        {
                System.out.println(“Outer Class”);
        }
```

```
static class B
{
    int b;
    B(int b)
    {
    int c = b+10;
    this.b = c;
    }
    void show()
    {
            // print();  INVALID
            A a1 = new A(10);
            a1.print();
            System.out.println("b="+b);
            display();
    } // End of Method
} // End of class B
} // End of class A
```

Static-Nested Class Cannot Invoke
The Object Methods of Outer class Directly

Static-Nested Class : Example 1 …


```
class Test
{
        public static void main(String args[])
        {
                A.B b1 = new A.B(10);
                b1.show();
        }
}
```

Output

```
a=10
b=20
Outer Class
```

Non-Static Nested Class

- Non-Static-Nested classes do not have ***static*** keyword applied
- Non-Static-Nested classes can use the instance fields/methods of the outer class directly.
- To create an object for the non-static nested class, use this syntax:

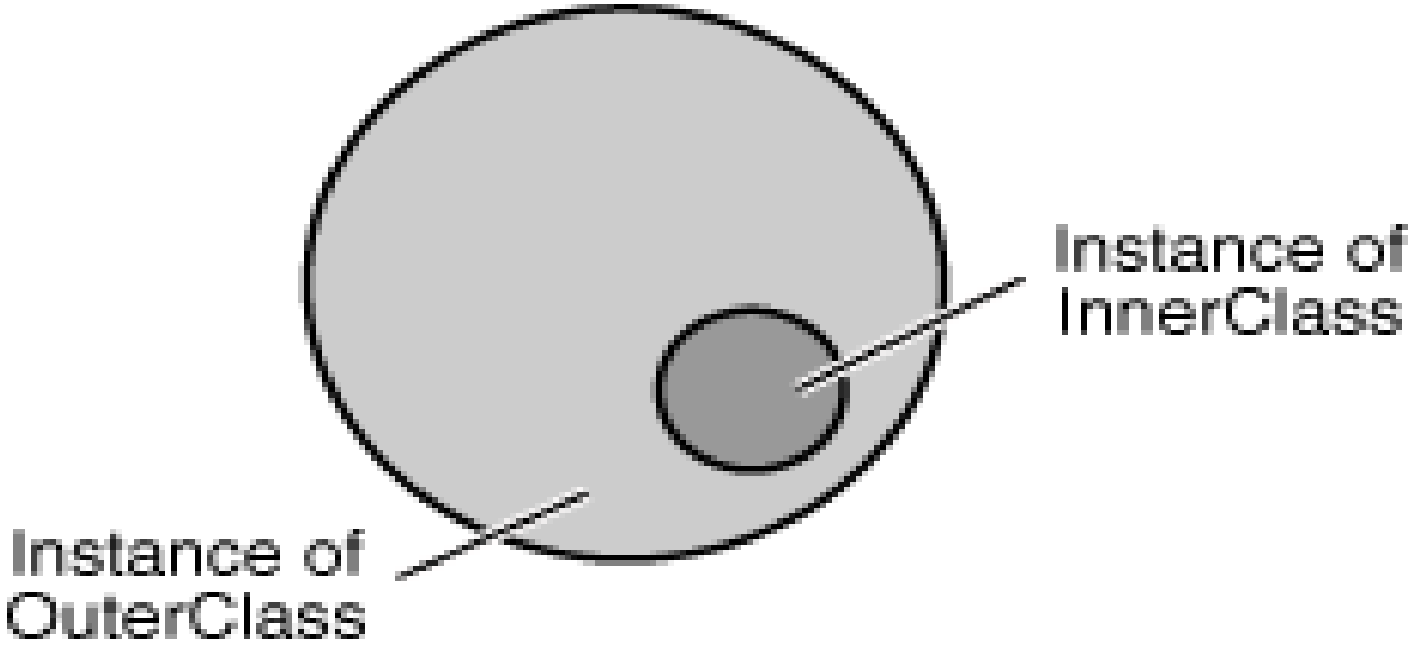
<Outer-Class-Name>.<Nested-Class-Name> Nested-Class-Object-Reference
=
Outer-Class-Object-Reference.new Nested-Class-Name();

OR

<Outer-Class-Name>.<Nested-Class-Name> Nested-Class-Object-Reference
=
Outer-Class-Object-Reference.new Nested-Class-Name(<parameters>);

Non-Static Nested Class

- An Instance of Non-Static-Nested class can Exists Only Inside an instance of Outer-Class
- So, In order to Create an Instance of Non-Static-Nested class, You have to First Create an Instance of Outer-Class



Non-Static-Nested Class : Example 1


```
class A
{
        private int a;
        A(int a)
        {
        this.a =a;
        }
        void print()
        {
                System.out.println("a="+a);
        }
        static      void      display()
        {
                System.out.println(“Outer Class”);
        }
```

```
class B
{
      int b;
      B(int b)
      {
      int c = b+10;
      this.b = c;
      }
      void show()
      {
            print();
            A a1 = new A(10);
            display();
      }
} // End of class B
} // End of class A
```

Non-Static-Nested Class Can Invoke The Object Methods of Outer class Directly

Non-Static-Nested Class : Example 1

...

```
class Test
{
        public static void main(String args[])
        {
                A.B b1 = new A(20).new B(10);
                b1.show();
                A a = new A(30);
                A.B b2 = a.new B(40);
                b2.show();
        }
}
```

```
a=20
Outer Class
a=30
Outer Class
```

Non-Static-Nested Class : Example 2

- Non-Static-Nested classes can not have static fields and Methods

```
// File Name : xyz.java
class A
{
        class B
        {
                private static int c=10;
                B()
                {
                System.out.println("Hello Inner Class");
                }
                static      void      display()
                {
                System.out.println("Static Method");
                }
        }// End of class inner class B
}// End of outer class A
```

```
xyz.java:6: inner classes cannot
have static declarations
        private static int c=10;
                          ^
xyz.java:11: inner classes cannot
have static declarations
        static  void    display()
                        ^
2 errors
```

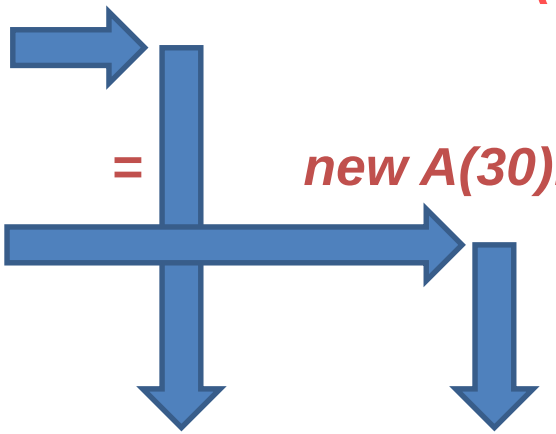
Non-Static-Nested Class : Example 3

```
class A
{
        private int a;
        private int b=10;
        A(int a)
        {
                this.a=a;
        }

        void show()
        {
        B b1 = new B(30);
        b1.show();
        } // End of Method
} // End of Outer class A
```

```
class B
{
        private int b;
        B(int b)
        {
                this.b =b;
        }
        void show()
        {
                int b=20;
                System.out.println("a="+a);
                System.out.println("b="+b);
                System.out.println("this.b="+this.b);
                System.out.println("Outer b="+A.this.b);
        }// End of Method
} // End of B inner class
```

Non-Static-Nested Class : Example 3 …

```
class InnerTest
{
        public static void main(String args[])
        {
                A       a1      =       new A(20);
                A.B     b1      =       a1.new B(-30);
                b1.show();

                A.B     b2      =       new A(30).new B(-40);
                b2.show();
        }
}
```



a=20	a=30
b=20	b=20
this.b=-30	this.b=-40
Outer b=10	Outer b=10

Thank You