#15

INNER or NESTED CLASSES

ZOOMING

Inner Class

- Inner class means one class which is a member of another class.
- There are basically four types of inner classes in java.
  - Nested Inner class.
  - Method Local inner classes.
  - Anonymous inner classes.
  - Static nested classes.

Nested Inner Class

- we can’t have static method in a nested inner class.
- Because an inner class is implicitly associated with an object of its outer class so it cannot define any static method for itself

```
class Outer {
    // Simple nested inner class
    class Inner {
        public void show() {
            System.out.println(“nested inner class");
        }
    }
}
class Main {
    public static void main(String[] args) {
        Outer.Inner in = new Outer().new Inner();
        in.show();
    }
}
```

Static Nested Class

Static nested classes are not technically an inner class. They are like a static member of outer class.

```
class Outer {
   private static void outerMethod() {
     System.out.println("inside outerMethod");
   }

   // A static inner class
   static class Inner {
     public static void main(String[] args) {
        System.out.println("inside inner class Method");
        outerMethod();
     }
   }

}
```

```
inside inner class
Method inside outerMethod
```

Method Local Inner Class

- Inner class can be declared within the method of an outer class.

```
class Outer {
    void outerMethod() {
        System.out.println("inside outerMethod");
        // Inner class is local to outerMethod()
        class Inner {
            void innerMethod() {
                System.out.println("inside innerMethod");
            }
        }
        Inner y = new Inner();
        y.innerMethod();
    }
}
class MethodDemo {
    public static void main(String[] args) {
        Outer x = new Outer();
        x.outerMethod();
    }
}
```

Method Local Inner Class

- Method Local inner classes can't use local variable of outer method until that local variable is not declared as final.

```
class Outer {
    void outerMethod() {
        int x = 98;
        System.out.println("inside outerMethod");
        class Inner {
            void innerMethod() {
                System.out.println("x= "+x);
            }
        }
        Inner y = new Inner();
        y.innerMethod();
    }
}
class MethodLocalVariableDemo {
    public static void main(String[] args) {
        Outer x=new Outer();
        x.outerMethod();
    }
}
```

```
local variable x is accessed from within inner class;
needs to be declared final
```

Method Local Inner Class

- Method Local inner classes can't use local variable of outer method until that local variable is not declared as final.

```
class Outer {
    void outerMethod() {
        final int x = 98;
        System.out.println("inside outerMethod");
        class Inner {
            void innerMethod() {
                System.out.println("x= "+x);
            }
        }
        Inner y = new Inner();
        y.innerMethod();
    }
}
class MethodLocalVariableDemo {
    public static void main(String[] args) {
        Outer x=new Outer();
        x.outerMethod();
    }
}
```

```
Inside outerMethod
X = 98
```

Anonymous Inner Class

Anonymous inner classes are declared without any name at all.

```
class Demo {
    void show() {
        System.out.println("super class");
    }
}
class Flavor1Demo {

    //  An anonymous class with Demo as base class
    static Demo d = new Demo() {
        void show() {
            super.show();
            System.out.println("sub class");
        }
    };
    public static void main(String[] args){
        d.show();
    }
}
```

```
Super class
Sub class
```