

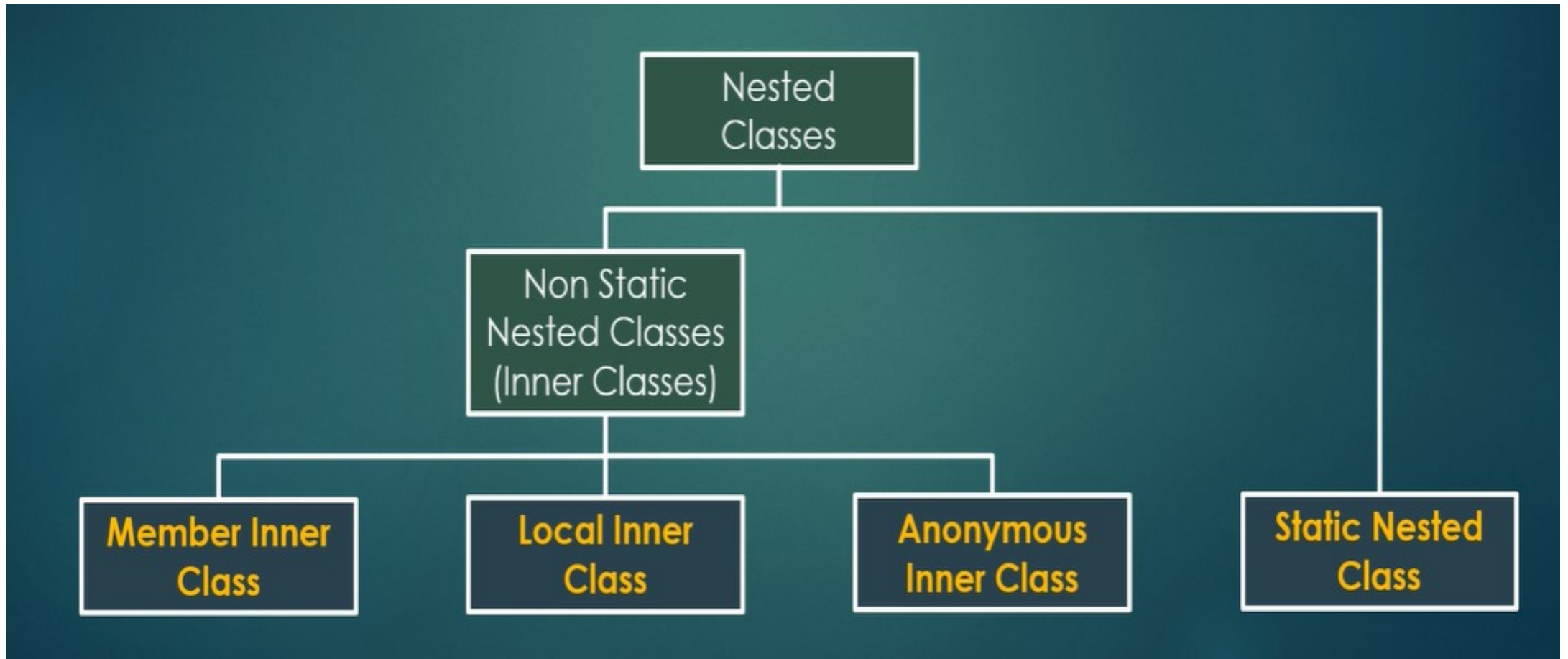
# Types of Nested Classes

Dr. Partha Pratim Sarangi  
School of Computer Engineering

# Nested Classes Overview

- A Nested Class is a class that is defined within another class.
- They enable you to logically group classes that are only used in one place, thus this increases the use of encapsulation, create more readable, maintainable code and it requires less code to write.
- Finally, it can provide Code Optimization.
- A Nested class can access all the members of the outer class, including private data members and methods.
- There are four types of nested classes (A nested class that is not static is called an inner class).

# Types of Nested Classes



# Types of Inner Classes

1. Non-static member inner class
  2. Static inner class
  3. Method local inner class
  4. Anonymous inner class
- Advantages of inner class
    - Logical grouping of classes
    - Increased encapsulation
    - More readable, maintainable code

# Member Inner Class

- Member Inner Class is a non-static class that is created inside a class but outside a method.
- Member Inner class can access the instance variable member of the Outer class.
- If you want to instantiate Inner class, first you must have to create the instance of Outer class.
- After that an instance of Inner class is created inside the instance of Outer class and with this, you can access the method inside Inner class.

# Non-static member inner class

- A class is defined within another class without using static modifier is treated as a non-static class.
- An inner class is declared by using any access modifier such as private, protected, public, and default.

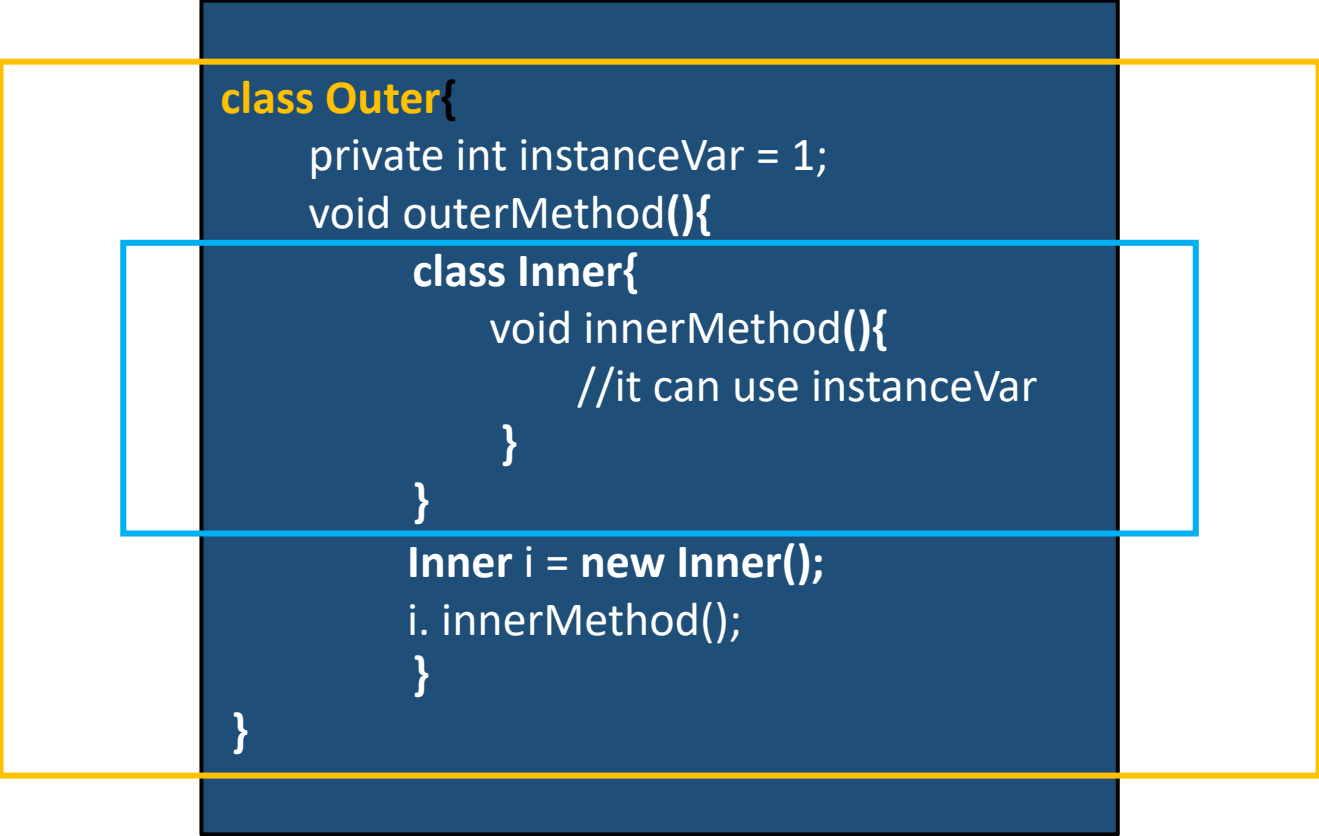
```
class Outer {  
    class Inner {  
        public void show()  
        {  
            System.out.println("Inside a nested  
class");  
        }  
    }  
}
```

```
class Main {  
    public static void main(String[] args)  
    {  
        //Outer obj = new Outer();  
        //Outer.Inner in = obj.new Inner();  
        Outer.Inner in = new Outer().new  
Inner();  
        in.show();  
    }  
}
```

# Local Inner Class

Local Inner Class is a non-static class that is created inside a class but (and this is the difference from the previous Member Inner Class) inside a method.

**Local Inner class** is accessing the **instance variable** member of the **Outer** class.



```
class Outer{  
    private int instanceVar = 1;  
    void outerMethod(){  
        class Inner{  
            void innerMethod(){  
                //it can use instanceVar  
            }  
        }  
        Inner i = new Inner();  
        i. innerMethod();  
    }  
}
```

The diagram illustrates a Local Inner Class. A large yellow rectangle represents the 'Outer' class. Inside it, a blue rectangle represents the 'outerMethod()' method. Within the blue rectangle, a smaller light blue rectangle represents the 'Inner' class. The code shows the 'Inner' class being defined inside the 'outerMethod()' method of the 'Outer' class. The 'Inner' class has a method 'innerMethod()' that can access the 'instanceVar' of the 'Outer' class. An instance of the 'Inner' class is created and its 'innerMethod()' is called within the 'outerMethod()' of the 'Outer' class.

You can instantiate Inner class inside of the method which contains the **Inner** Class. This way we can access the inner method.

```
Outer o = new Outer();  
o.outerMethod();
```

After that if you create an instance of **Outer** class, you can call the method which contains the **Inner** Class.

# Static inner class

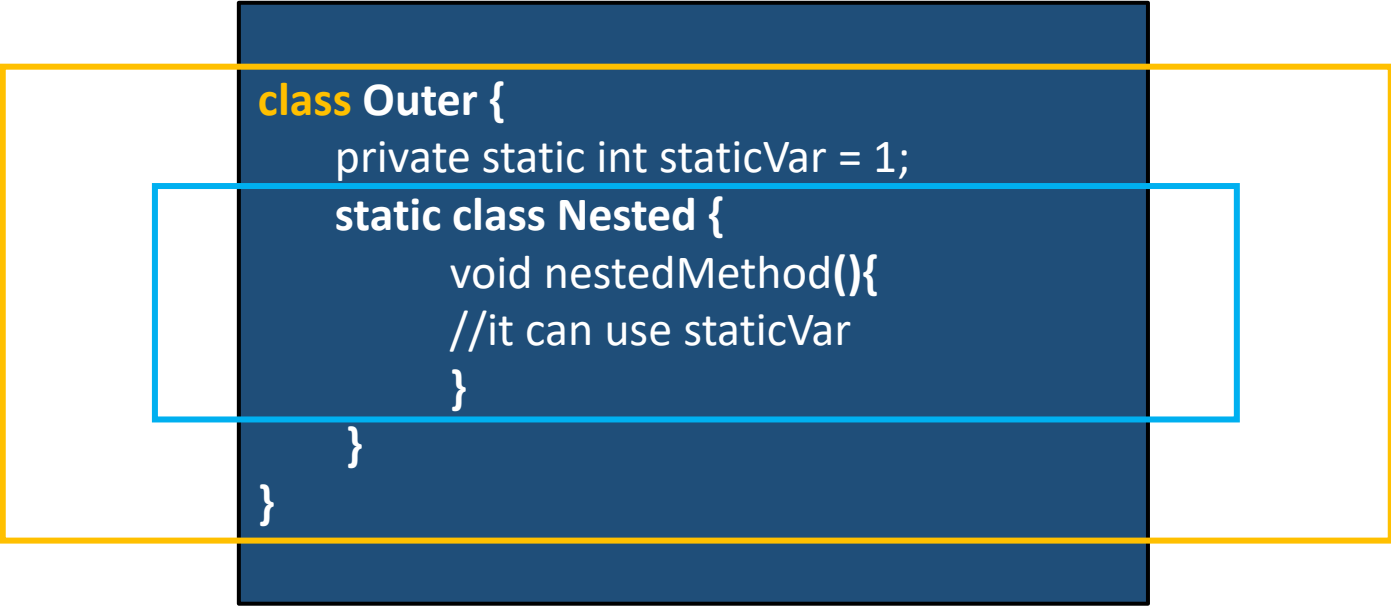
- A static in Java in general means it belongs to a class and not to the individual instances.
- So, a static member or method in Java need not be accessed with an object rather directly or using a class name.
- A static class in Java can contain only static members.
- Also, we cannot instantiate a static class.
- We can make a class static if and only if it is a nested class.



# Static Nested Class

A static nested class is a static class that is defined at the same level as static variables. It's a nested class which is a static member of the outer class.

Static **Nested** class is accessing the static variable member of the **Outer** class. But keep in mind that it cannot access non-static instance variables and methods of the outer class.



```
class Outer {  
    private static int staticVar = 1;  
    static class Nested {  
        void nestedMethod(){  
            //it can use staticVar  
        }  
    }  
}
```

**Nested** class can be accessed without instantiating the **Outer** class, using other static members.

```
Outer.Nested i = new Outer.Nested();  
i.nestedMethod();
```

After that you can access the method inside **Nested** class.

```
class Outer_Class {
    static int total; // static variable
    static void sum(int val1, int val2) { //
static method
        System.out.print("Static method to
calculate sum:" + " ");
        total = val1 + val2;
        System.out.println(val1 + "+" +
val2); // print the numbers
    }
    static class Inner_Class { // static class
public void displaySum() {
        sum(25, 75); // call static method
        // print the value in static variable
total, it holds the sum of two numbers
        System.out.println("Sum of two
numbers:" + total);
    }
}
```

```
    }
}

public class Main {
    public static void main(String args[]) {
        // declare static class object
        Outer_Class.Inner_Class obj = new
Static_Class.Nested_Class();
        obj.displaySum(); // call displaySum
method inside a static class
    }
}
```

# Anonymous Inner Class

- A class that has no name is known as an anonymous inner class in Java.
- It is a type of inner class that is the same as local inner class but the only difference is that the class has no class name and a single object is created of the class.
- It makes the code more concise. It is used if we want to use the local class once.
- The main advantage of anonymous inner class is we create the object of abstract class and interface.

```
public abstract class Test
{
    public abstract void show( );
    void fun( )
    {
        System.out.println("Good Morning");
    }
    public static void main(String[] args)
    {
        Test t = new Test( )
        {
            public void show( )
            {
                System.out.println("Welcome to Java Programming");
            }
        }
        t.fun( );
        t.show( );
    }
}
```

# How to create an object of the abstract class

```
abstract class Person{  
    abstract void eat();  
}
```

```
class TestAnonymousInnerClass {  
    public static void main(String[] args){  
        Person p = new Person(){  
            void eat(){  
                System.out.println("Nice fruits");  
            }  
        }  
        p.eat();  
    }  
}
```

# Key Points to Remember

- Java treats the inner class as a member of a class. They are just like variables and methods declared inside a class.
- Since inner classes are members of the outer class, you can apply any access modifiers like private, protected to your inner class which is not possible in normal classes, whereas default, public access modifiers can be used outer and inner classes.
- Using the nested class will make your code more readable and provide better encapsulation.
- Non-static nested classes (inner classes) have access to other members of the outer/enclosing class, even if they are declared private.