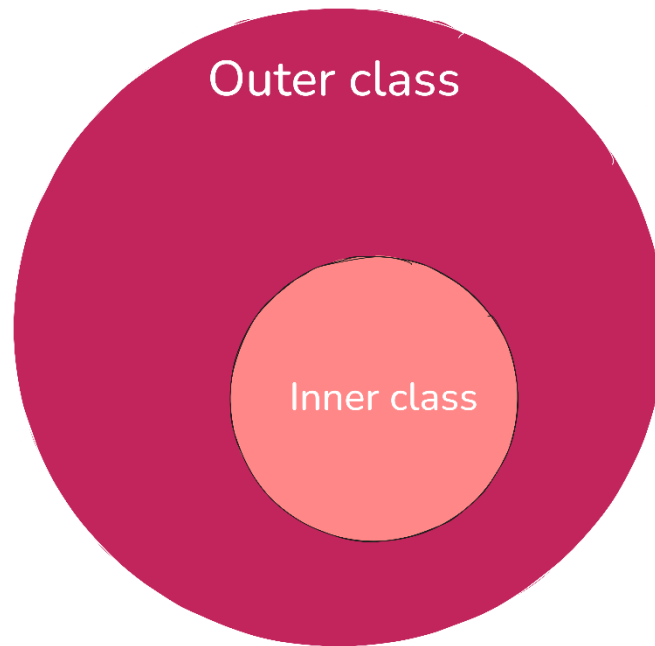


• 10.2-Inner Class

➤ **Introduction**

In Java, an **inner class** refers to a class that is declared **inside another class**. Inner classes were primarily introduced to group logically related classes together, reflecting the object-oriented nature of Java and its attempt to mirror real-world relationships.



➤ **Syntax of Inner Class**

The general syntax for defining an inner class is as follows:

```
class OuterClass {  
    // Outer class code  
  
    class InnerClass {  
        // Inner class code  
    }  
}
```

➤ **Advantages of Inner Classes**

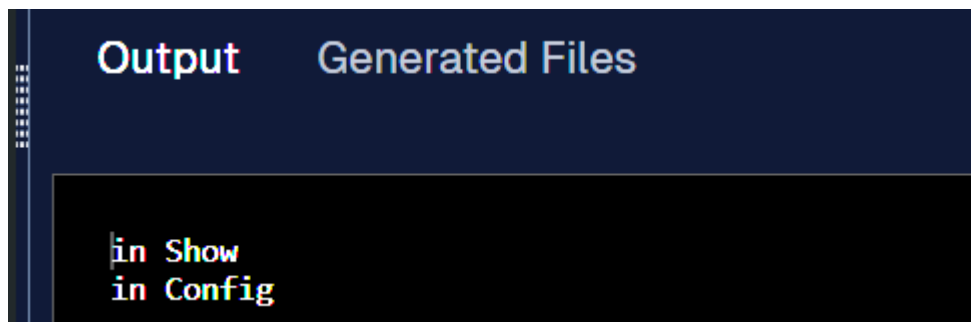
Using inner classes has several advantages:

- **Encapsulation**: It allows better encapsulation of the code, especially when the inner class is logically dependent on the outer class.
- **Access to Outer Class Members**: Inner classes can access the private members (fields and methods) of the outer class, providing an added dimension to the class design.
- **Cleaner Code**: It leads to more modular, readable, and optimized code.

➤ Example: Basic Inner Class

```
class A {  
    public void show() {  
        System.out.println("in Show");  
    }  
  
    class B {  
        public void config() {  
            System.out.println("in Config");  
        }  
    }  
}  
  
public class Demo {  
    public static void main(String[] args) {  
        A obj = new A(); // Creating an instance of the outer class  
        obj.show();      // Calling method of the outer class  
  
        // Creating an instance of the inner class using the outer class object  
        A.B obj1 = obj.new B();  
        obj1.config();   // Calling method of the inner class  
    }  
}
```

Output:



```
Output  Generated Files  
  
in Show  
in Config
```

Explanation

In this example:

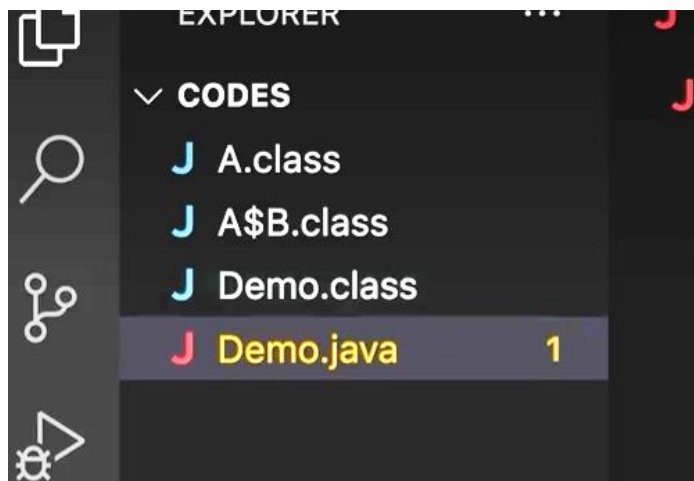
- We have two classes, A (outer class) and B (inner class).
- Class A has a method show(), and class B has a method config().
- To create an object of the inner class B, we first need to create an object of the outer class A. This is because the inner class is dependent on the outer class.

➤ Code Explanation:

```
A.B obj1 = obj.new B();
```

Here, A.B indicates that B is an inner class of A, and obj.new B() uses the object obj of A to create an instance of B.

After executing we get two class files one of class A and the other Other of class B.



Here, B class file is created with the name **A\$B** where \$ represents B class is inner class and it belongs to class A.

➤ Inner Class Dependency

An inner class is usually tightly coupled with the outer class. Without the outer class, the inner class often makes no sense. In the example above, **class B depends on class A** for its existence.

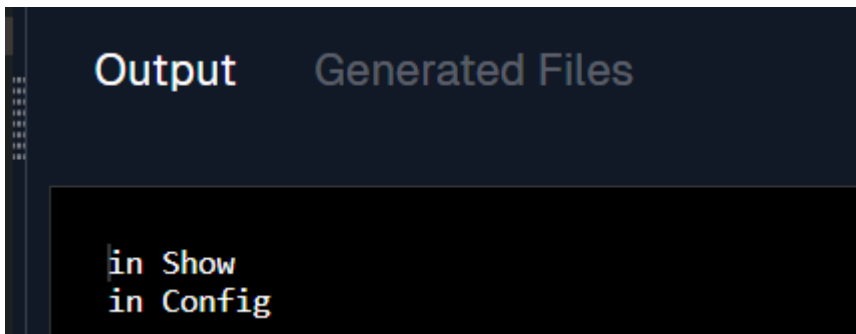
➤ Static Inner Class

Inner classes can also be marked as static. A **static inner class** is independent of the outer class and can be accessed without creating an object of the outer class.

➤ Example: Static Inner Class

```
class A {  
    public void show() {  
        System.out.println("in Show");  
    }  
  
    static class B {  
        public void config() {  
            System.out.println("in Config");  
        }  
    }  
}  
  
public class Demo {  
    public static void main(String[] args) {  
        A obj = new A(); // Creating an instance of the outer class  
        obj.show();      // Calling method of the outer class  
  
        // Creating an instance of the static inner class without an outer class object  
        A.B obj1 = new A.B();  
        obj1.config();   // Calling method of the static inner class  
    }  
}
```

Output:



```
Output    Generated Files  
  
in Show  
in Config
```

➤ Key Points for Static Inner Class:

- **Independent of the Outer Class:** A static inner class can be instantiated without an object of the outer class.
- **Static Context:** The static inner class behaves like a regular class and cannot access the non-static members of the outer class directly.
- **Rules for Inner and Outer Classes**

- **Static Modifier:** Only inner classes can be marked as static. Outer classes cannot be declared as static, and any attempt to do so will result in a compile-time error.
 - **Allowed Modifiers for Outer Class:** The valid access modifiers for an outer class are public, final, and abstract.
-

➤ **Summary**

- Inner classes are declared within another class.
- An inner class can access private members of its outer class.
- To instantiate an inner class, an object of the outer class is required.
- A static inner class can be instantiated without creating an object of the outer class.
- Static modifier applies only to inner classes, not outer classes.