Inner class in java

Inner class means one class which is a member of another class. There are basically four types of inner classes in java.

1) Nested Inner class
2) Anonymous inner classes
2) Method Local inner classes
4) Static nested classes

1. Nested Inner Class

■ Nested inner class syntax:

```
class Outer{
    class Inner{
    }
}
```

■ Creating inner class object inside outer class:

```
class Outer{
    class Inner{
        void print(){
            System.out.println("Inner print");
        }
    }

    void print(){
        System.out.println("Outer print");

        // Create an object of inner class
        Inner inner = new Inner();

        // Access inner class members
        inner.print();
    }

    public static void main(String[] args) {
        Outer outerObj = new Outer();
        outerObj.print();
        // Output:
        // Outer print
        // Inner print
    }
}
```

■ Creating inner class object outside outer class or in static method:

```
class Outer{

    class Inner{
        void print(){
            System.out.println("Hello");
        }
    }

    public static void main(String[] args) {
        Outer outerObj = new Outer();

        // Creating inner class object
        Outer.Inner innerObj = outerObj.new Inner();

        innerObj.print();
        // Output: Hello
    }
}
```

■ **Inner Class Shadowing:** If a Java inner class declares fields or methods with the same names as field or methods in its enclosing class, the inner fields or methods are said to shadow over the outer fields or methods. Here is an example:

```
class Outer{
    private String text = "Outer text";

    class Inner{
        private String text = "Inner text";
        void print(){
            System.out.println(text);

            // Accessing outer class's shadowed field
            System.out.println(Outer.this.text);
        }
    }

    public static void main(String[] args) {
        Outer outerObj = new Outer();
        Outer.Inner innerObj = outerObj.new Inner();

        innerObj.print();
        // Output:
        // Inner text
        // Outer text
    }
}
```

2. Anonymous inner classes

■ Anonymous classes in Java are nested classes without a class name. They are typically declared as either subclasses of an existing class, or as implementations of some interface.

■ Anonymous class example by extending a class:

```
class SuperClass{
    public void doIt(){
        System.out.println("Super Class doIt()");
    }
}

class TestAnonymous{
    public static void main(String[] args) {
        SuperClass instance = new SuperClass(){
            public void doIt(){
                System.out.println("Anonymous Class doIt()");
            }
        };

        instance.doIt();
        // Prints: Anonymous Class doIt()
    }
}
```

■ Anonymous class example by extending a class:

```
interface MyInterface{
    void doIt();
}

class TestAnonymous{
    public static void main(String[] args) {
        MyInterface instance = new MyInterface(){
            public void doIt(){
                System.out.println("Anonymous Class doIt()");
            }
        };

        instance.doIt();
        // Prints: Anonymous Class doIt()
    }
}
```

3. Method Local inner classes

```
class LocalInner{
    private int data = 20;

    void display(){
        int value = 50;

        class Local{
            void msg(){
                System.out.println(value + data);
            }
        }

        Local l = new Local();
        l.msg();
    }


    public static void main(String[] args) {
        LocalInner obj = new LocalInner();
        obj.display(); // Output: 70
    }
}
```

4. Static nested classes

■ Outer classes cannot be static, but nested/inner classes can be. That basically helps you to use the nested/inner class without creating an instance of the outerclass.

```
class Outer{
    public static class Nested{

    }

    public static void main(String[] args) {
        Outer.Nested obj = new Outer.Nested();
    }
}
```

■ Inner class(or non-static nested class) can access both static and non-static members of Outer class. A static class cannot access non-static members of the Outer class. It can access only static members of Outer class.

```
class Outer{
    static int data = 30;

    static class StaticInner {
        void msg(){
            System.out.println("Data is: " + data);
        }

        static void msg(String msg){
            System.out.println(msg);
        }
    }

    public static void main(String[] args) {
        Outer.StaticInner obj = new Outer.StaticInner();

        obj.msg(); // prints "Data is: 30"

        Outer.StaticInner.msg("Hello!"); // prints "Hello"
    }
}
```