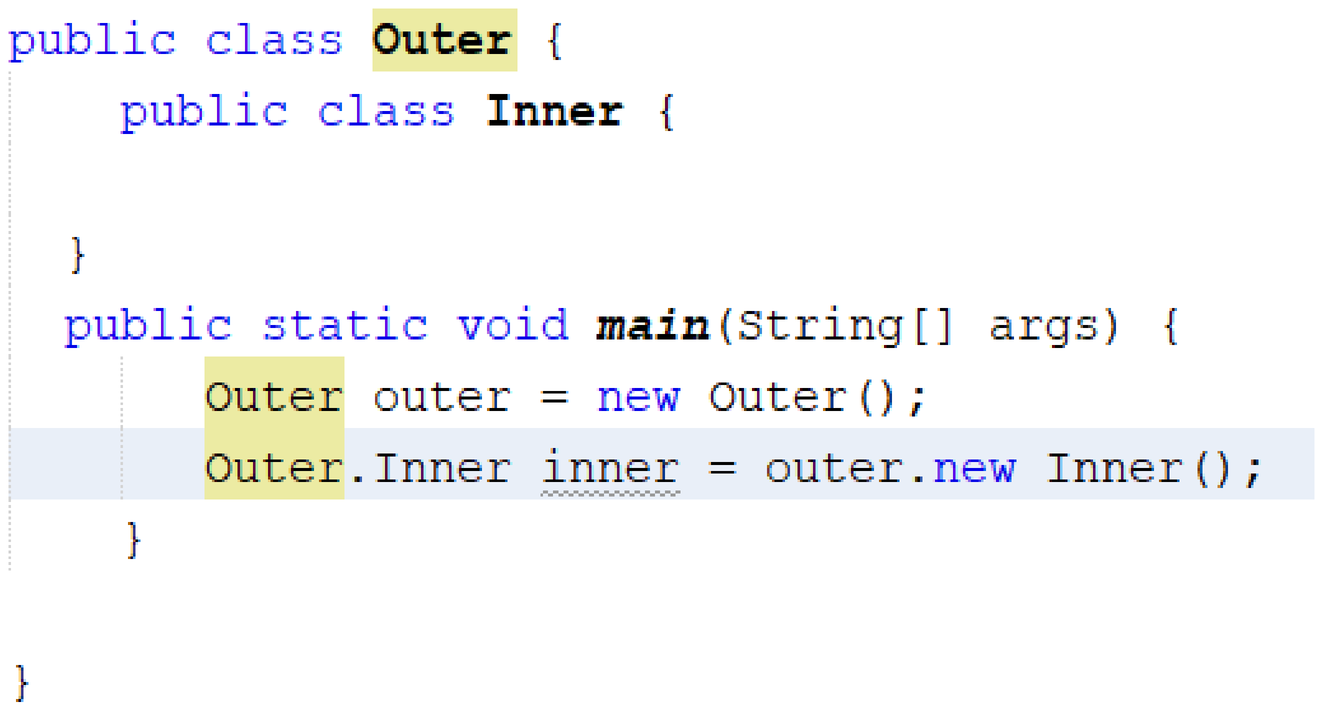
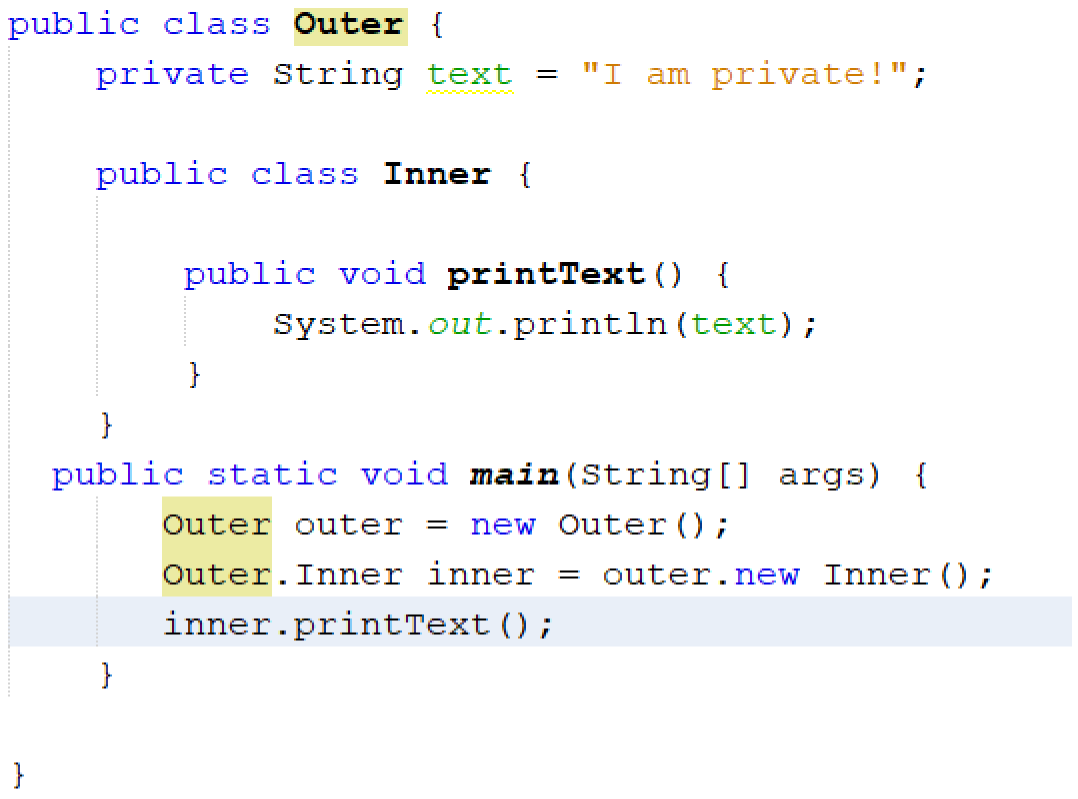
Non-static Nested Classes (Inner Classes)



```
public class Outer {
    public class Inner {

  }
  public static void main(String[] args) {
       Outer outer = new Outer();
       Outer.Inner inner = outer.new Inner();
    }

}
```

✓ Non-static nested classes (inner classes) have access to the fields of the enclosing class, even if they are declared private. Here is an example of that:



```java
public class Outer {
    private String text = "I am private!";

    public class Inner {

        public void printText() {
            System.out.println(text);
        }
    }
  public static void main(String[] args) {
       Outer outer = new Outer();
       Outer.Inner inner = outer.new Inner();
       inner.printText();
    }

}
```

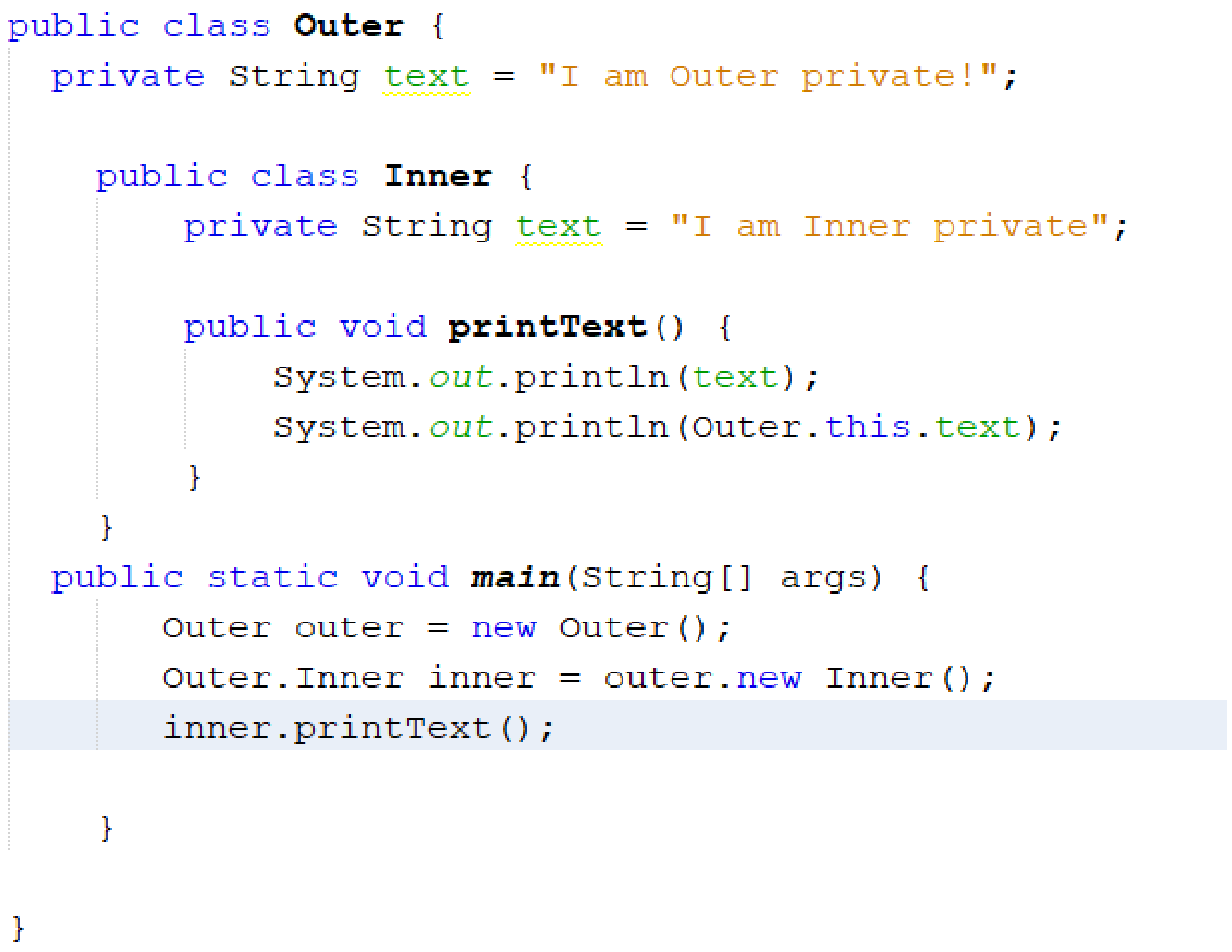
✓ To access inner class's member from outer class, you need to access via object of inner class.

```
public class Outer {
    public void show(){
        System.out.println("Show method");
        Inner inner = new Inner();
        inner.msg(); // Can access private member of inner class
        System.out.println("--End of Show method--");
        }
        class Inner{
        private void msg(){
        System.out.println("Inner Method");
        }
    }
  public static void main(String[] args) {
       Outer outer = new Outer();
       outer.show();
       Outer.Inner inner = outer.new Inner();
       inner.msg();
    }

}
```

Output:

```
Show method
Inner Method
--End of Show method--
Inner Method
```

- ✓ **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:

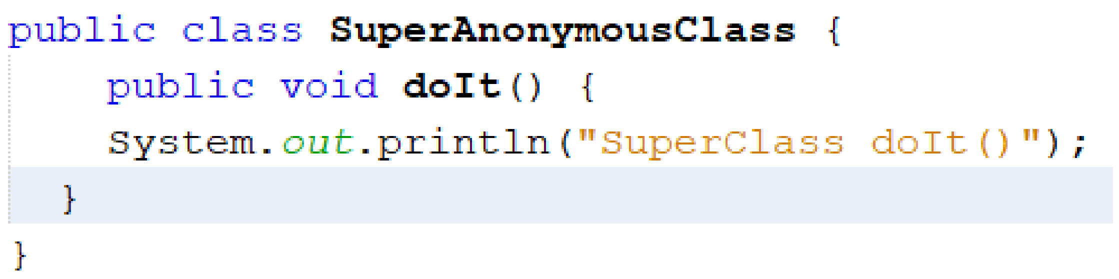


```java
public class Outer {
    private String text = "I am Outer private!";

        public class Inner {
            private String text = "I am Inner private";

            public void printText() {
                System.out.println(text);
                System.out.println(Outer.this.text);
            }
        }
    public static void main(String[] args) {
        Outer outer = new Outer();
        Outer.Inner inner = outer.new Inner();
        inner.printText();

    }

}
```

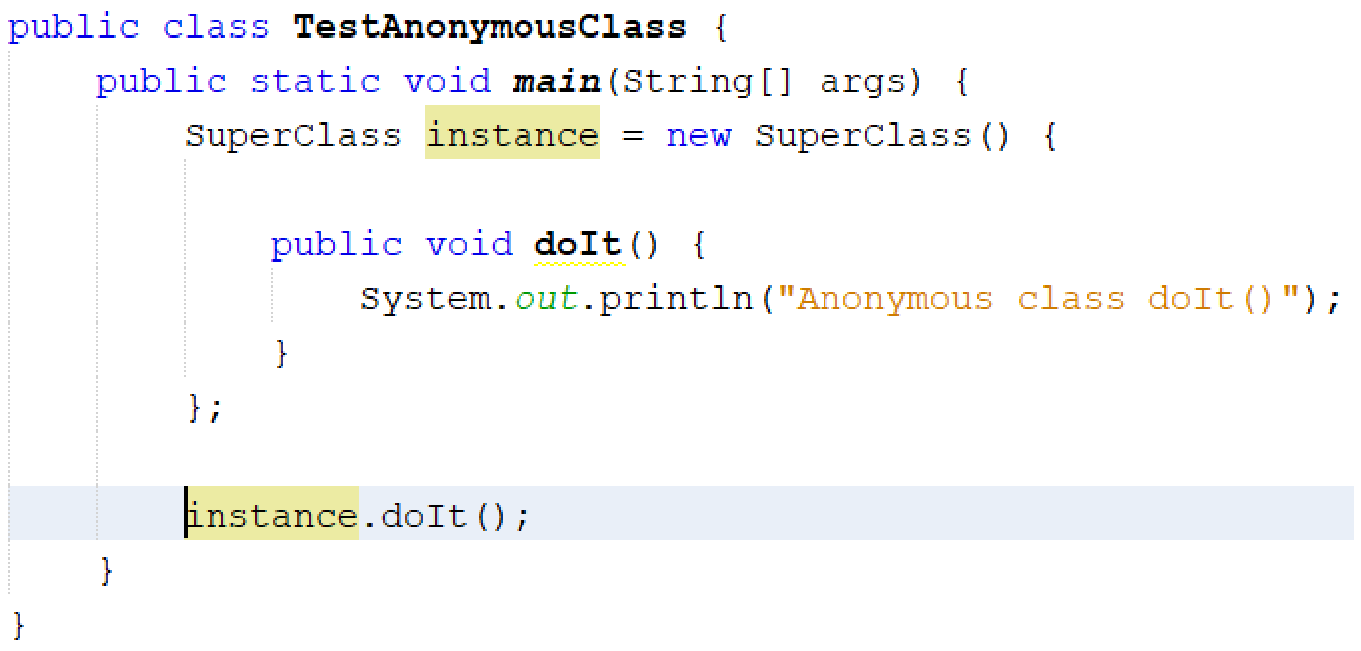
Anonymous Classes

- ✓ 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

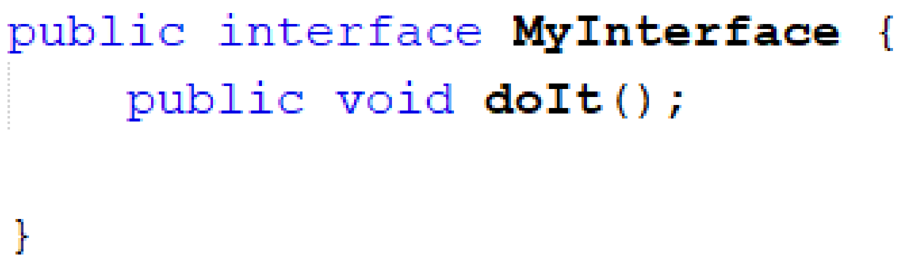


```
public class SuperAnonymousClass {
    public void doIt() {
    System.out.println("SuperClass doIt()");
  }
}
```

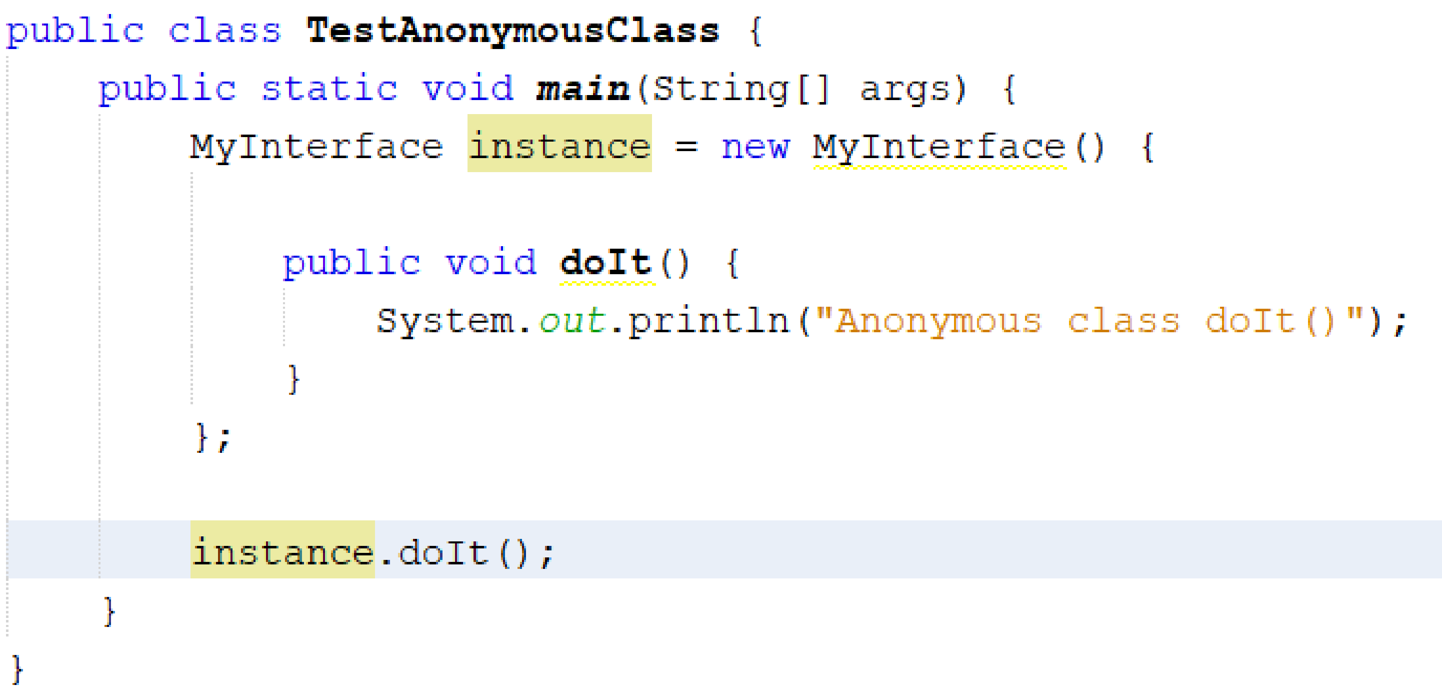


```
public class TestAnonymousClass {
    public static void main(String[] args) {
        SuperClass instance = new SuperClass() {

            public void doIt() {
                System.out.println("Anonymous class doIt()");
            }
        };

        instance.doIt();
    }
}
```

ANONYMOUS CLASS – EXAMPLE BY IMPLEMENTING AN INTERFACE

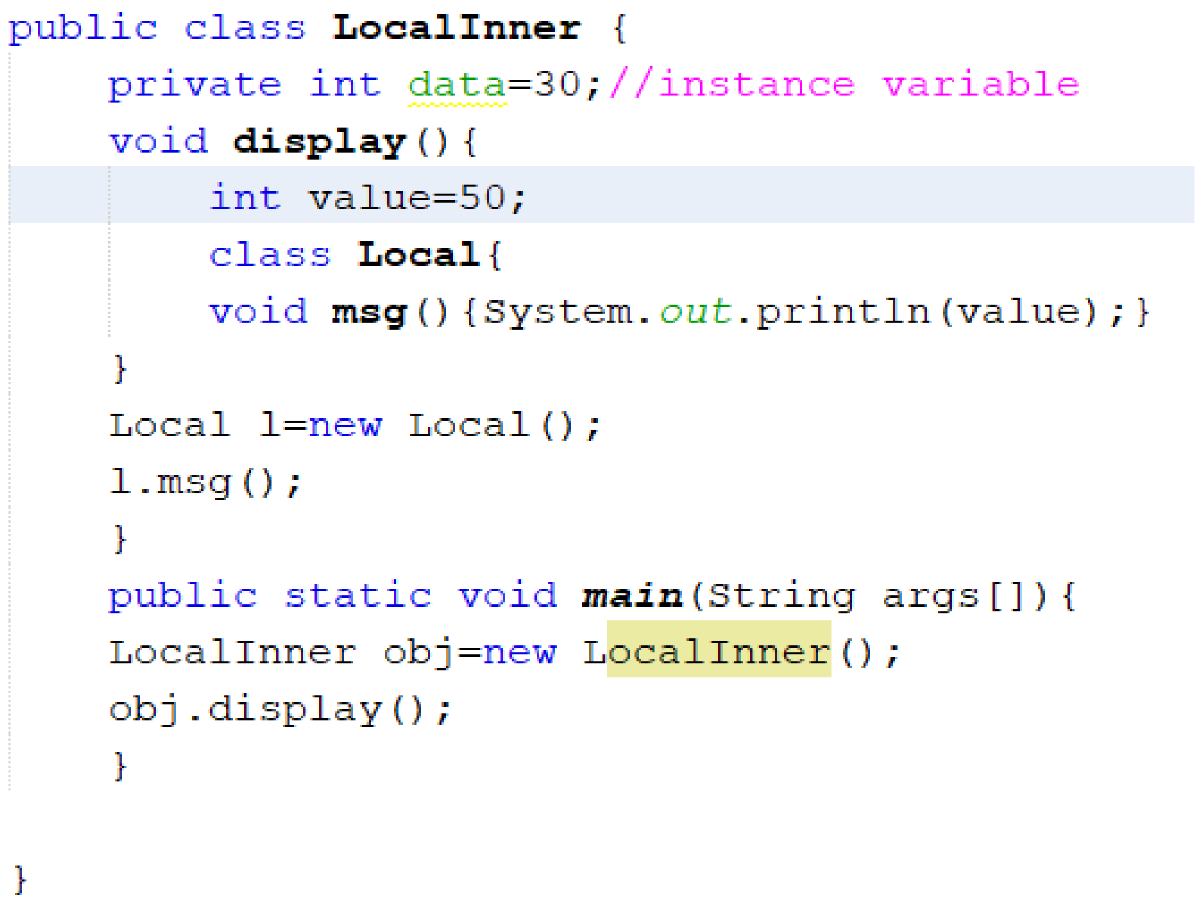


```java
public interface MyInterface {
	public void doIt();


}
```



```java
public class TestAnonymousClass {
	public static void main(String[] args) {
		MyInterface instance = new MyInterface() {

			public void doIt() {
				System.out.println("Anonymous class doIt()");
			}
		};

		instance.doIt();
	}
}
```

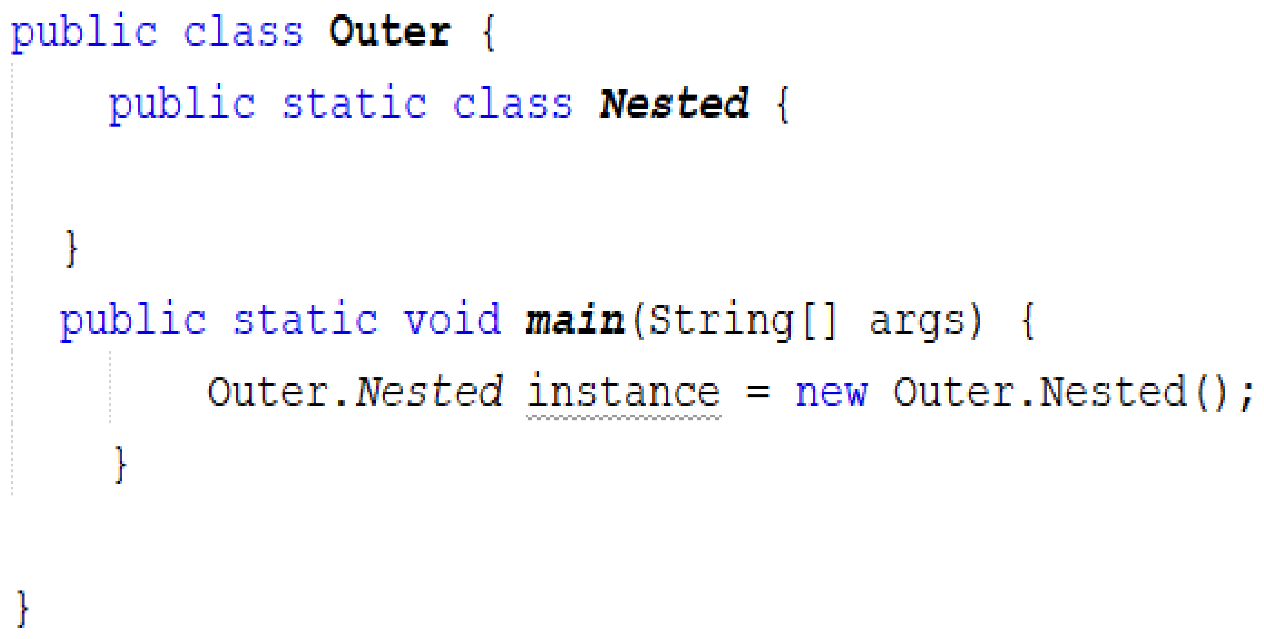
LOCAL INNER CLASS



```
public class LocalInner {
    private int data=30;//instance variable
    void display(){
        int value=50;
        class Local{
        void msg(){System.out.println(value);}
    }
    Local l=new Local();
    l.msg();
    }
    public static void main(String args[]){
    LocalInner obj=new LocalInner();
    obj.display();
    }

}
```

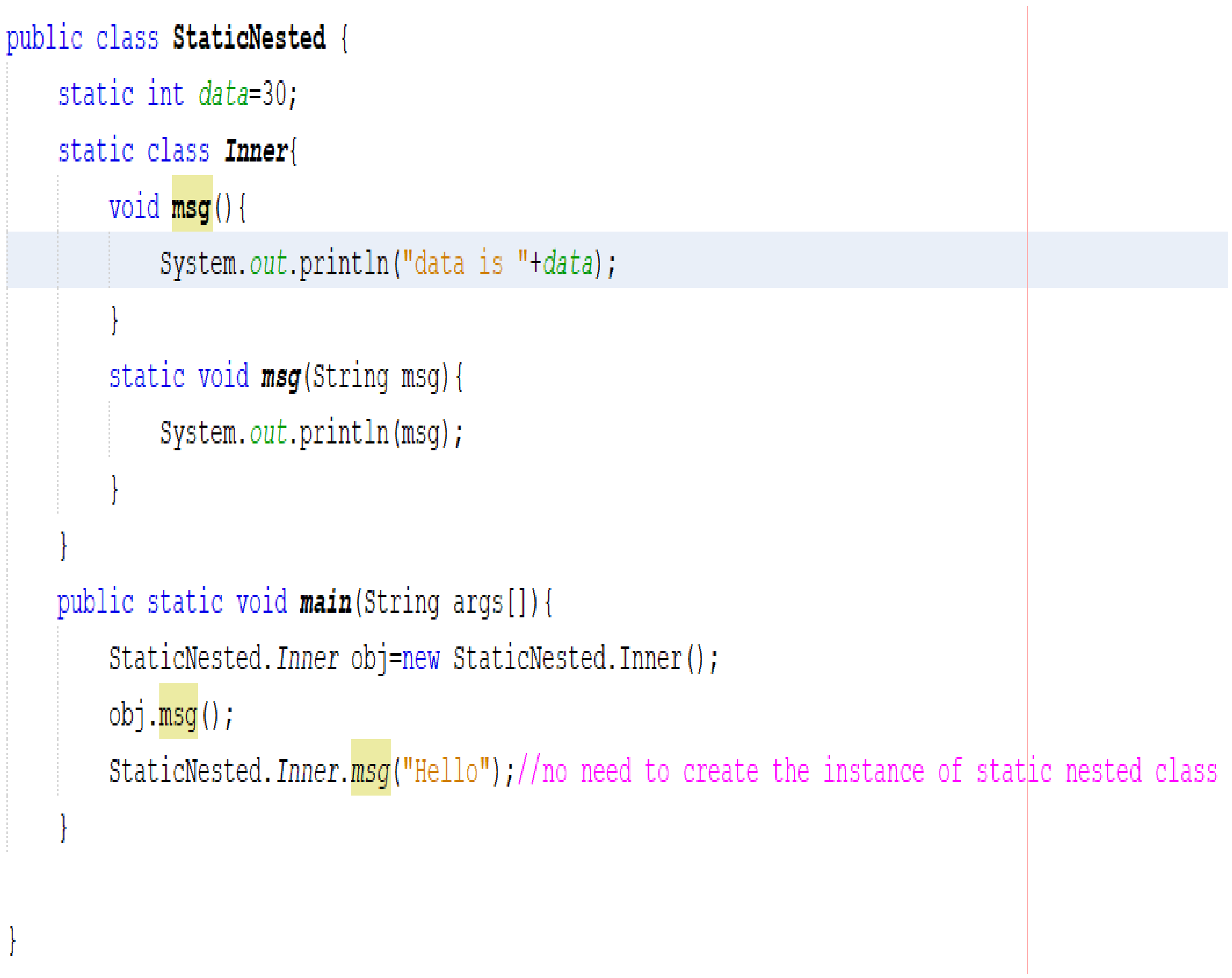
Static Nested Classes

- ✓ 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.



```java
public class Outer {
    public static class Nested {


  }
  public static void main(String[] args) {
        Outer.Nested instance = 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.



```java
public class StaticNested {
    static int data=30;
    static class Inner{
        void msg(){
            System.out.println("data is "+data);
        }
        static void msg(String msg){
            System.out.println(msg);
        }
    }
    public static void main(String args[]){
        StaticNested.Inner obj=new StaticNested.Inner();
        obj.msg();
        StaticNested.Inner.msg("Hello");//no need to create the instance of static nested class
    }

}
```