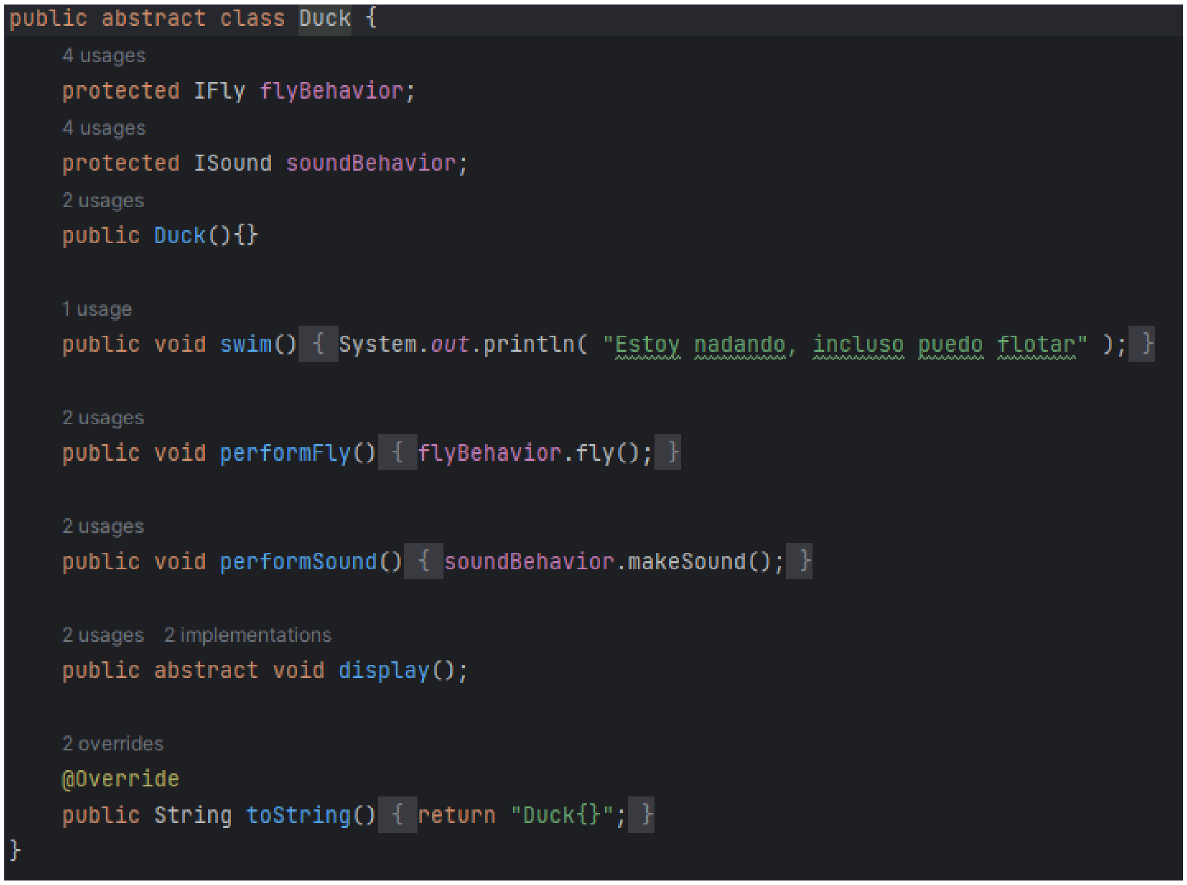
Patrón Estrategia

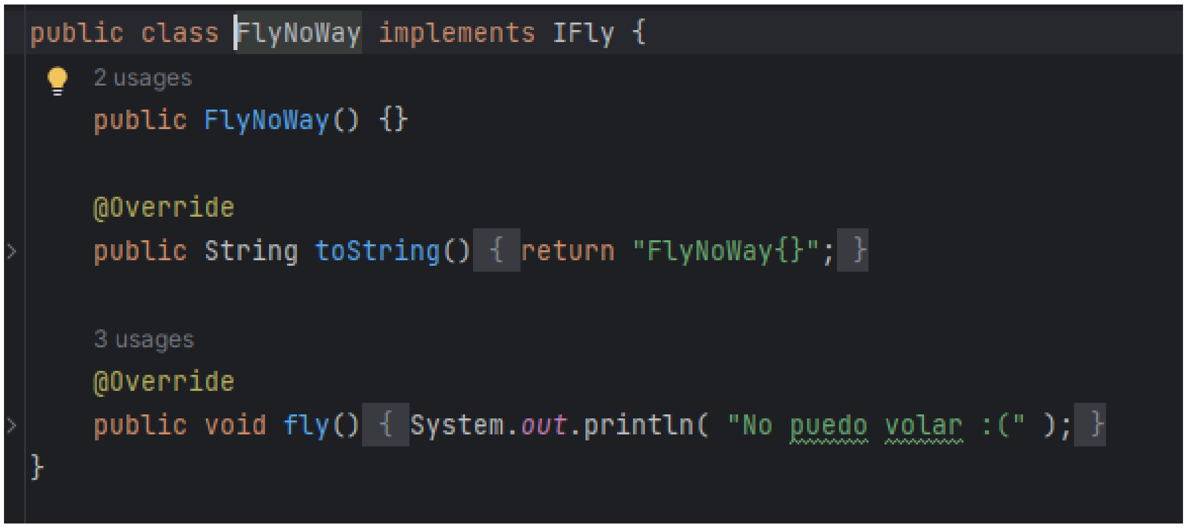
Nombre: Júnior Yesmín Morales Estrada **Carné:** 3490-20-12328

Clase Duck



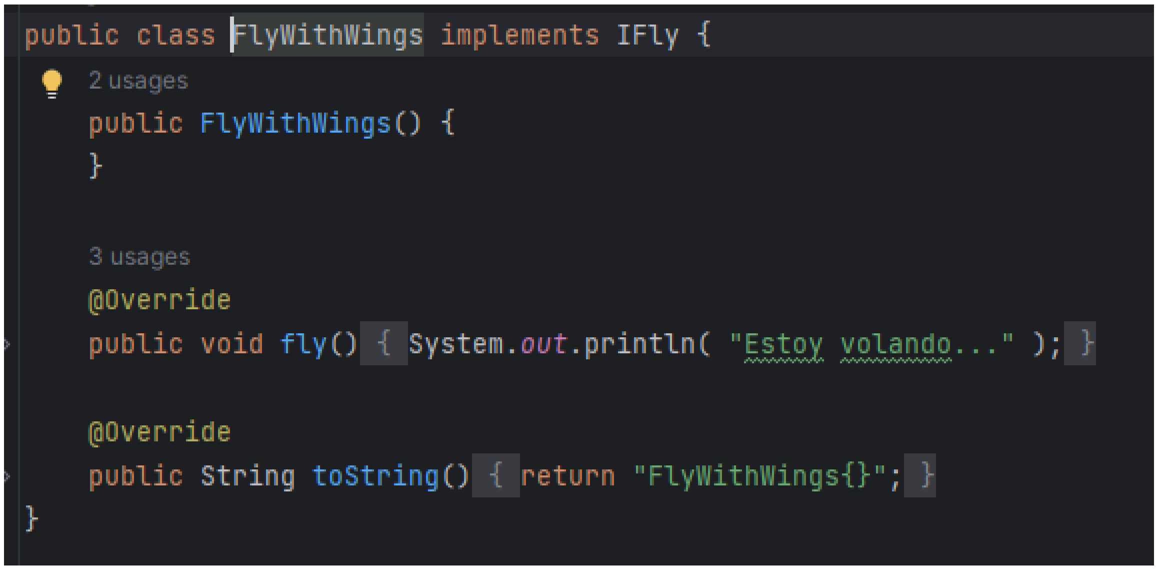
```
public abstract class Duck {
    4 usages
    protected IFly flyBehavior;
    4 usages
    protected ISound soundBehavior;
    2 usages
    public Duck(){}

    1 usage
    public void swim() { System.out.println( "Estoy nadando, incluso puedo flotar" ); }

    2 usages
    public void performFly() { flyBehavior.fly(); }

    2 usages
    public void performSound() { soundBehavior.makeSound(); }

    2 usages   2 implementations
    public abstract void display();

    2 overrides
    @Override
    public String toString() { return "Duck{}"; }
}
```

Clase FlyNoWay



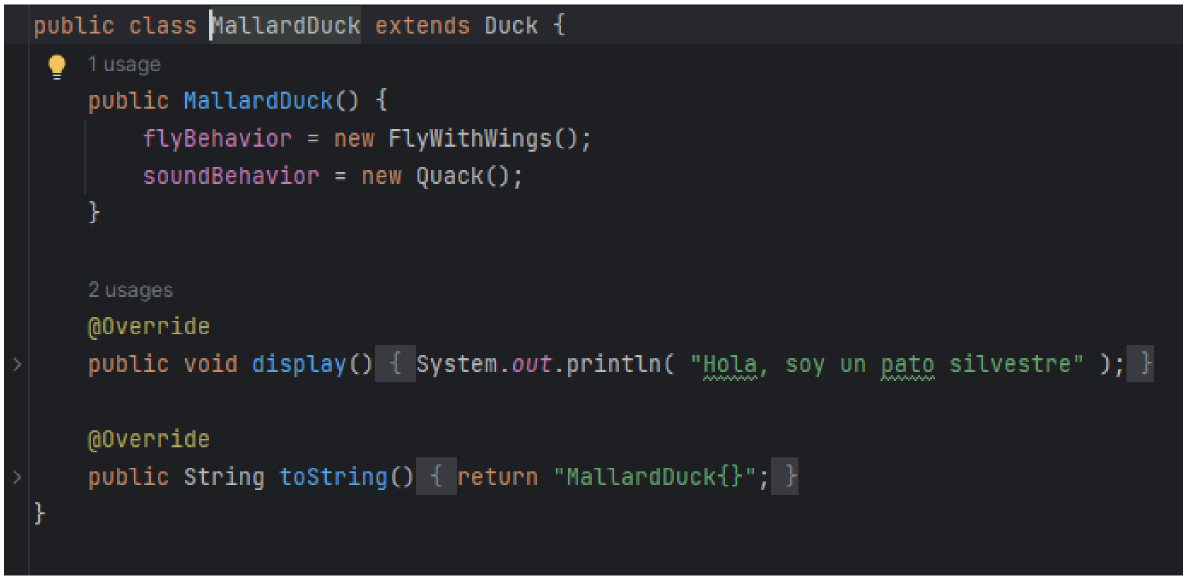
```
public class FlyNoWay implements IFly {
    2 usages
    public FlyNoWay() {}

    @Override
    public String toString() { return "FlyNoWay{}"; }

    3 usages
    @Override
    public void fly() { System.out.println( "No puedo volar :(" ); }
}
```

FlyWithWings



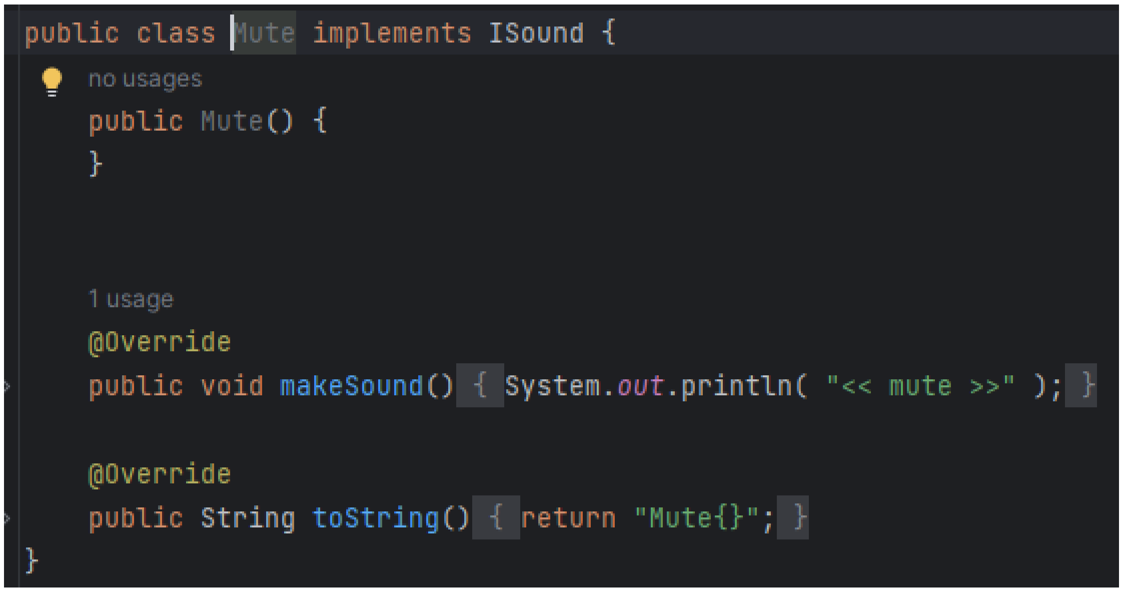
```
public class FlyWithWings implements IFly {
    2 usages
    public FlyWithWings() {
    }

    3 usages
    @Override
    public void fly() { System.out.println( "Estoy volando..." ); }

    @Override
    public String toString() { return "FlyWithWings{}"; }
}
```

MallardDuck



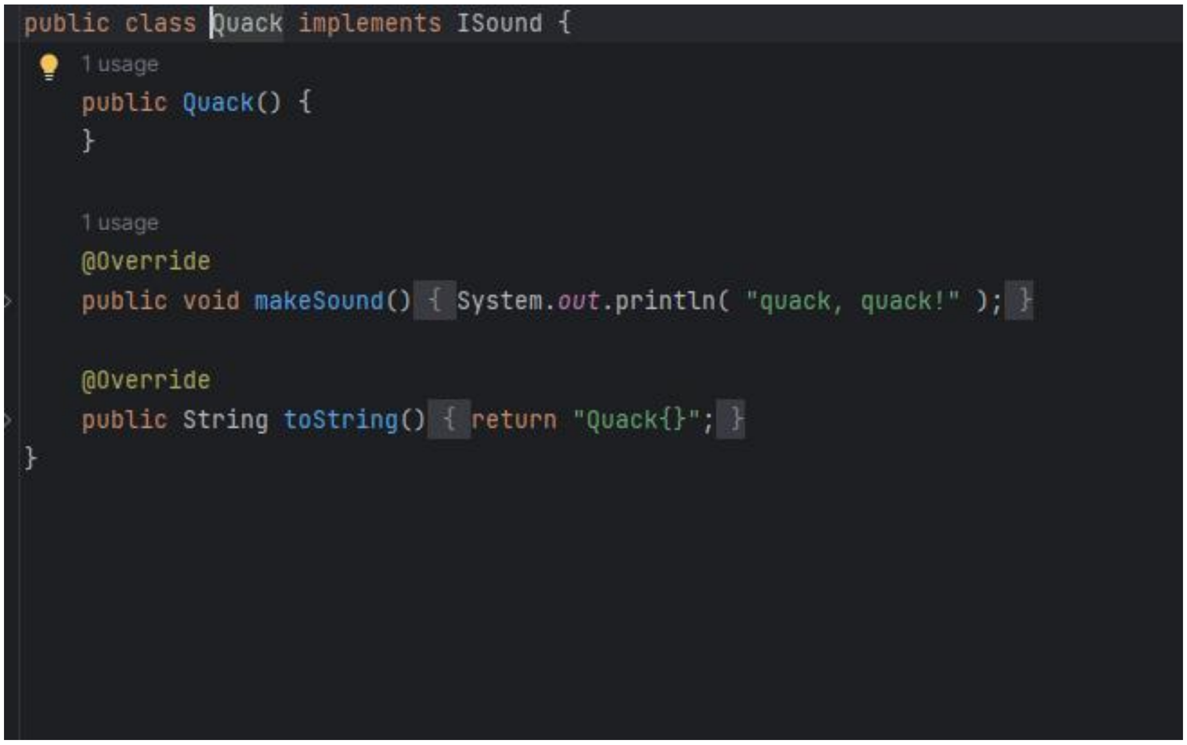
```
public class MallardDuck extends Duck {
    1 usage
    public MallardDuck() {
        flyBehavior = new FlyWithWings();
        soundBehavior = new Quack();
    }

    2 usages
    @Override
    public void display() { System.out.println( "Hola, soy un pato silvestre" ); }

    @Override
    public String toString() { return "MallardDuck{}"; }
}
```

Mute



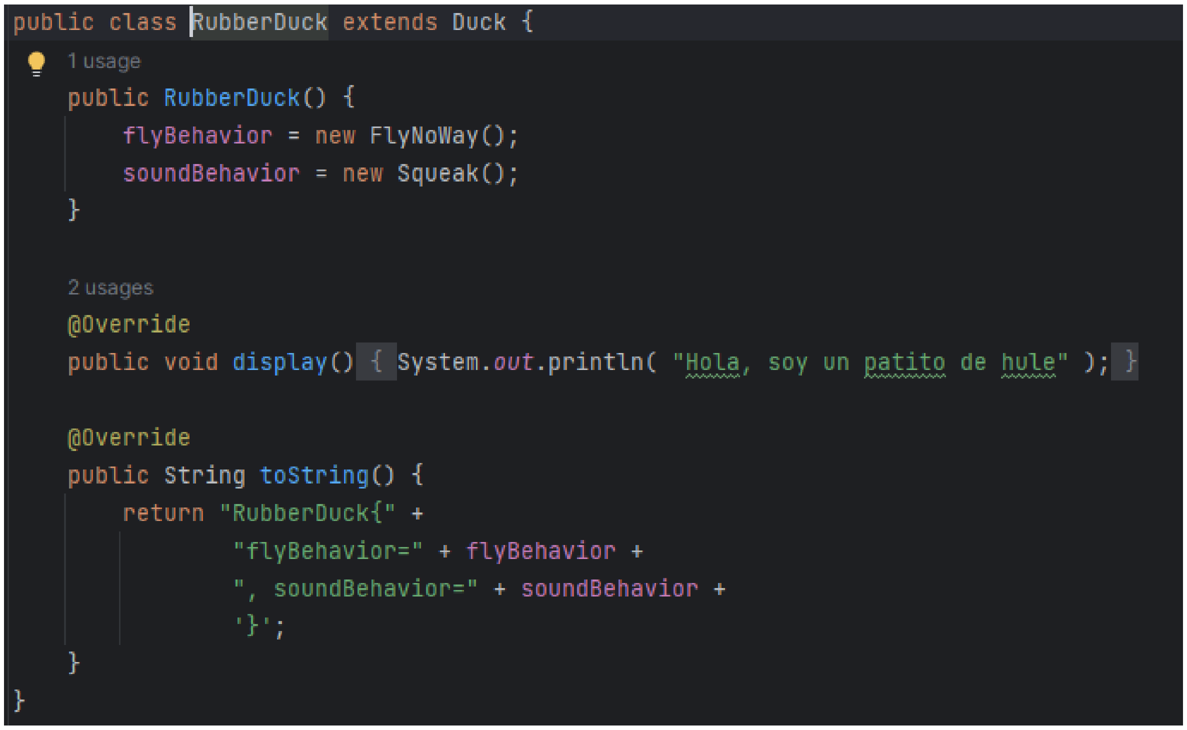
```
public class Mute implements ISound {
    no usages
    public Mute() {
    }


    1 usage
    @Override
    public void makeSound() { System.out.println( "<< mute >>" ); }

    @Override
    public String toString() { return "Mute{}"; }
}
```

Quack



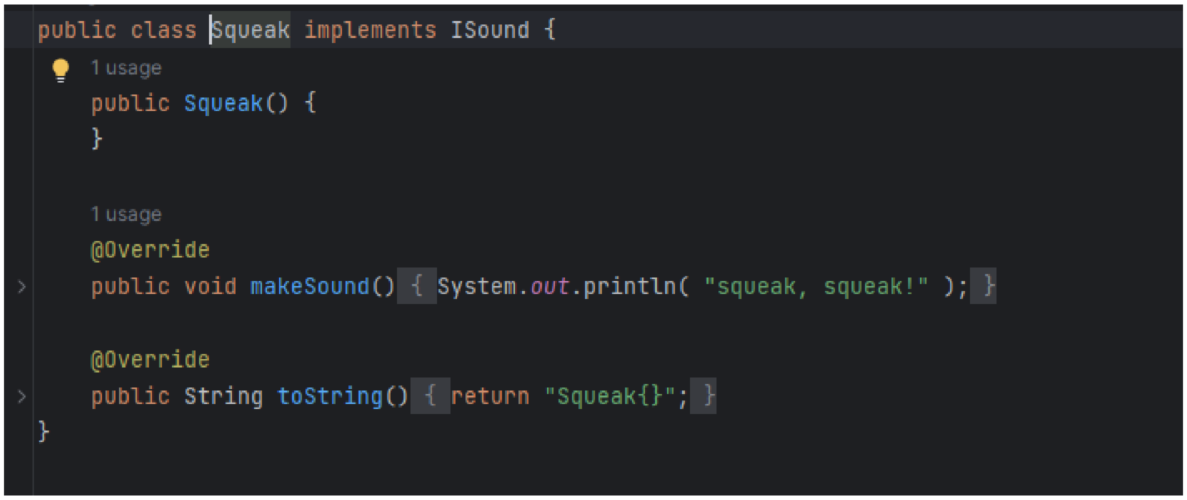
```
public class Quack implements ISound {
    1 usage
    public Quack() {
    }

    1 usage
    @Override
    public void makeSound() { System.out.println( "quack, quack!" ); }

    @Override
    public String toString() { return "Quack{}"; }
}
```

RubberDuck



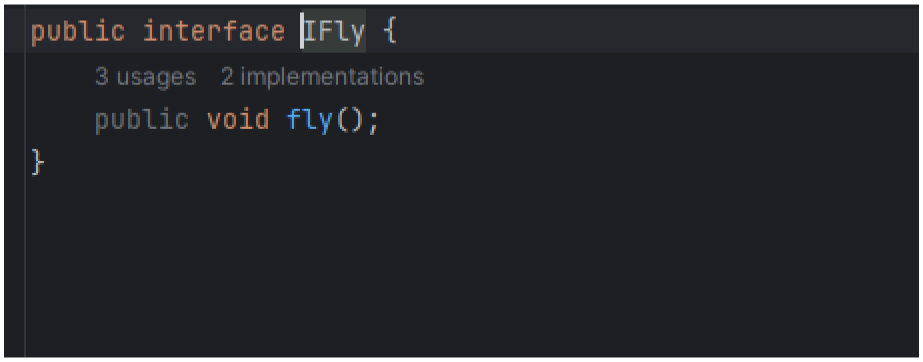
```
public class RubberDuck extends Duck {
    1 usage
    public RubberDuck() {
        flyBehavior = new FlyNoWay();
        soundBehavior = new Squeak();
    }

    2 usages
    @Override
    public void display() { System.out.println( "Hola, soy un patito de hule" ); }

    @Override
    public String toString() {
        return "RubberDuck{" +
                "flyBehavior=" + flyBehavior +
                ", soundBehavior=" + soundBehavior +
                '}';
    }
}
```

Squeak



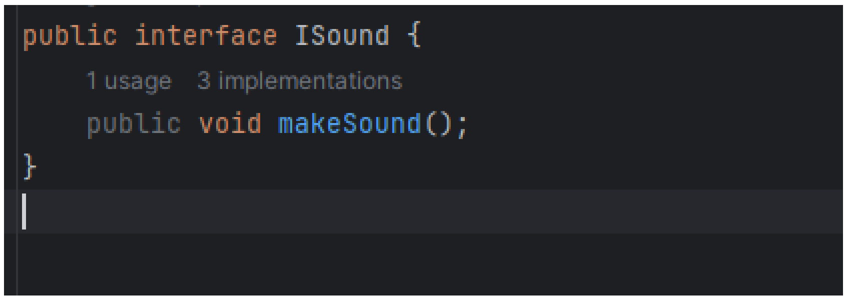
```
public class Squeak implements ISound {
    1 usage
    public Squeak() {
    }

    1 usage
    @Override
    public void makeSound() { System.out.println( "squeak, squeak!" ); }

    @Override
    public String toString() { return "Squeak{}"; }
}
```

Interface IFly



```
public interface IFly {
    3 usages   2 implementations
    public void fly();
}
```

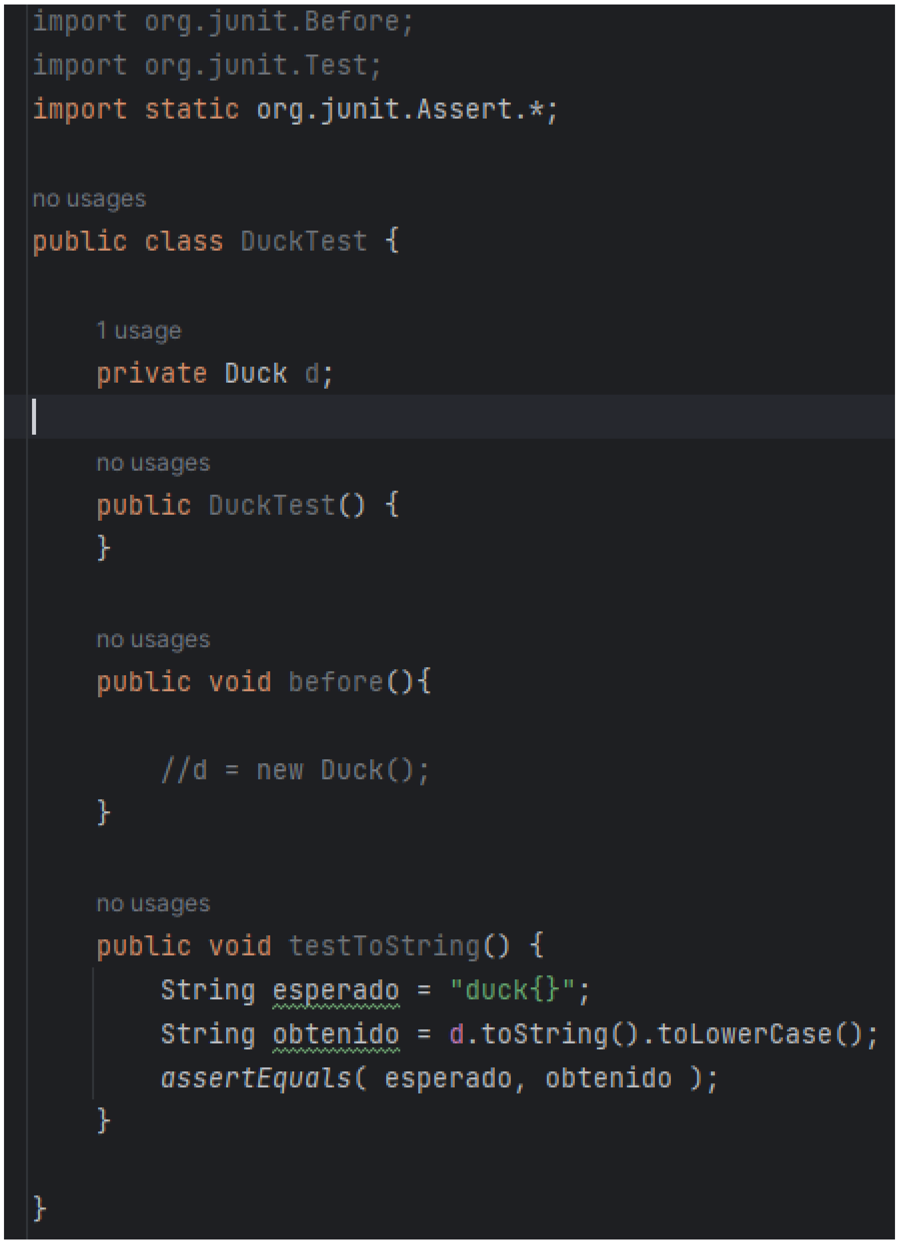
Interface



```
public interface ISound {
    1 usage   3 implementations
    public void makeSound();
}
```

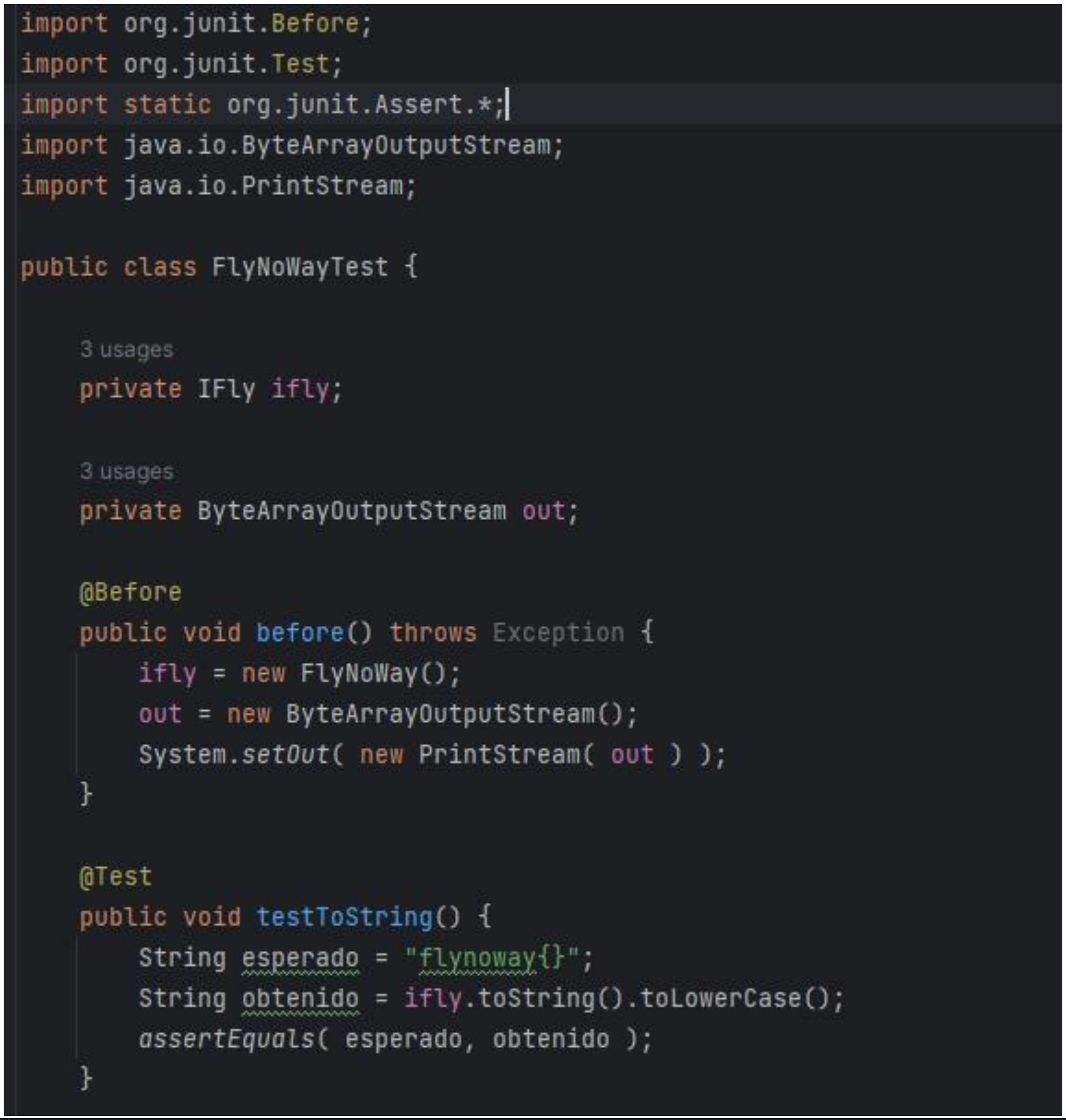
Test

DuckTest

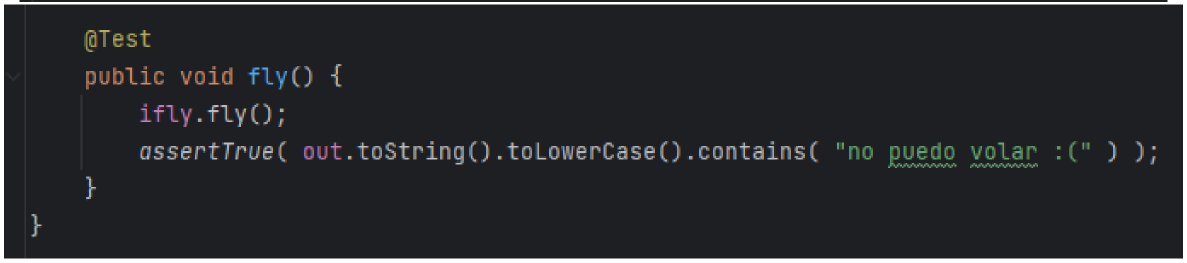


```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

no usages
public class DuckTest {

    1 usage
    private Duck d;

    no usages
    public DuckTest() {
    }

    no usages
    public void before(){

        //d = new Duck();
    }

    no usages
    public void testToString() {
        String esperado = "duck{}";
        String obtenido = d.toString().toLowerCase();
        assertEquals( esperado, obtenido );
    }

}
```

FlyNoWayTest

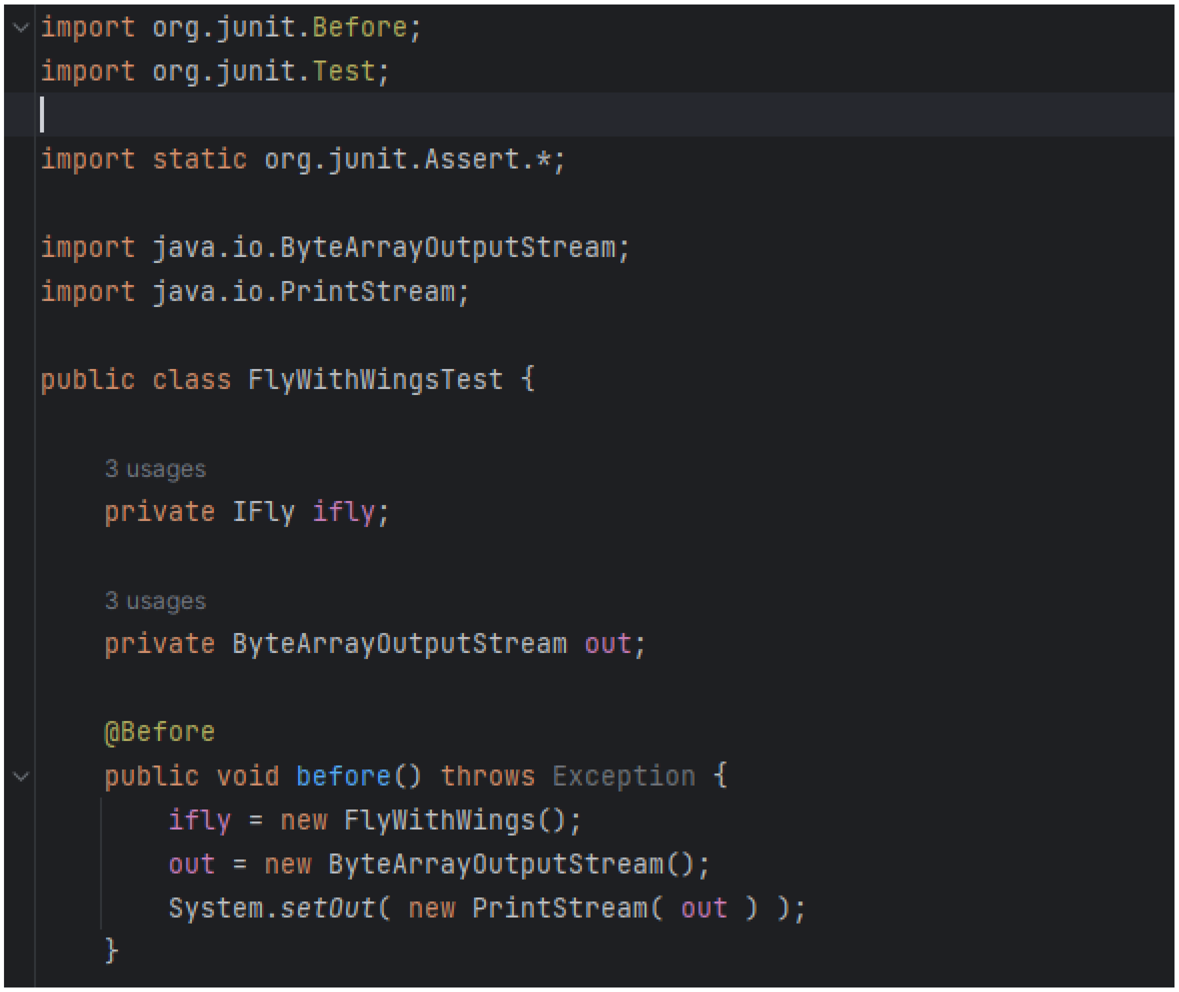


```java
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
import java.io.ByteArrayOutputStream;
import java.io.PrintStream;

public class FlyNoWayTest {

    3 usages
    private IFly ifly;

    3 usages
    private ByteArrayOutputStream out;

    @Before
    public void before() throws Exception {
        ifly = new FlyNoWay();
        out = new ByteArrayOutputStream();
        System.setOut( new PrintStream( out ) );
    }

    @Test
    public void testToString() {
        String esperado = "flynoway{}";
        String obtenido = ifly.toString().toLowerCase();
        assertEquals( esperado, obtenido );
    }
```

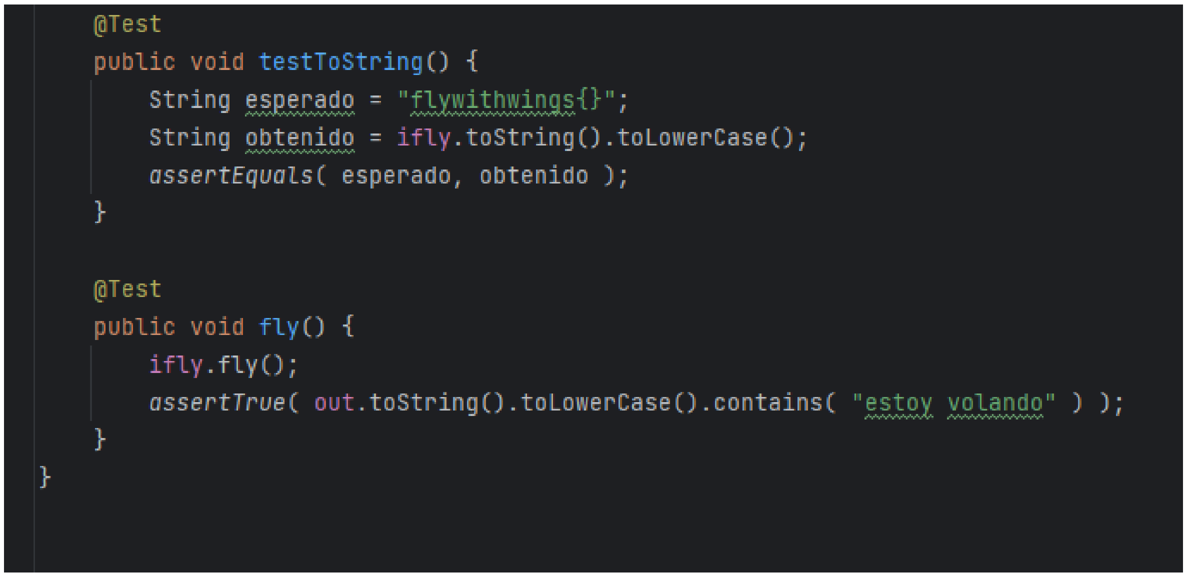


```java
    @Test
    public void fly() {
        ifly.fly();
        assertTrue( out.toString().toLowerCase().contains( "no puedo volar :(" ) );
    }
}
```

FlyWithWingsTest

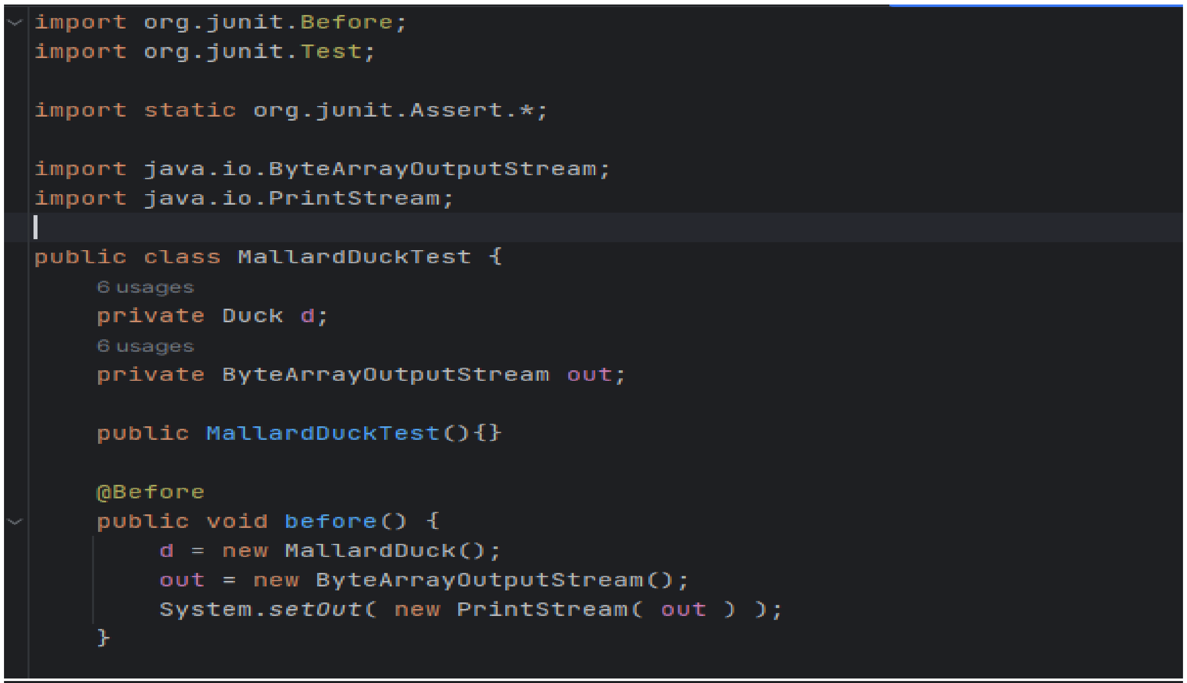


```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

import java.io.ByteArrayOutputStream;
import java.io.PrintStream;

public class FlyWithWingsTest {

    3 usages
    private IFly ifly;

    3 usages
    private ByteArrayOutputStream out;

    @Before
    public void before() throws Exception {
        ifly = new FlyWithWings();
        out = new ByteArrayOutputStream();
        System.setOut( new PrintStream( out ) );
    }
```

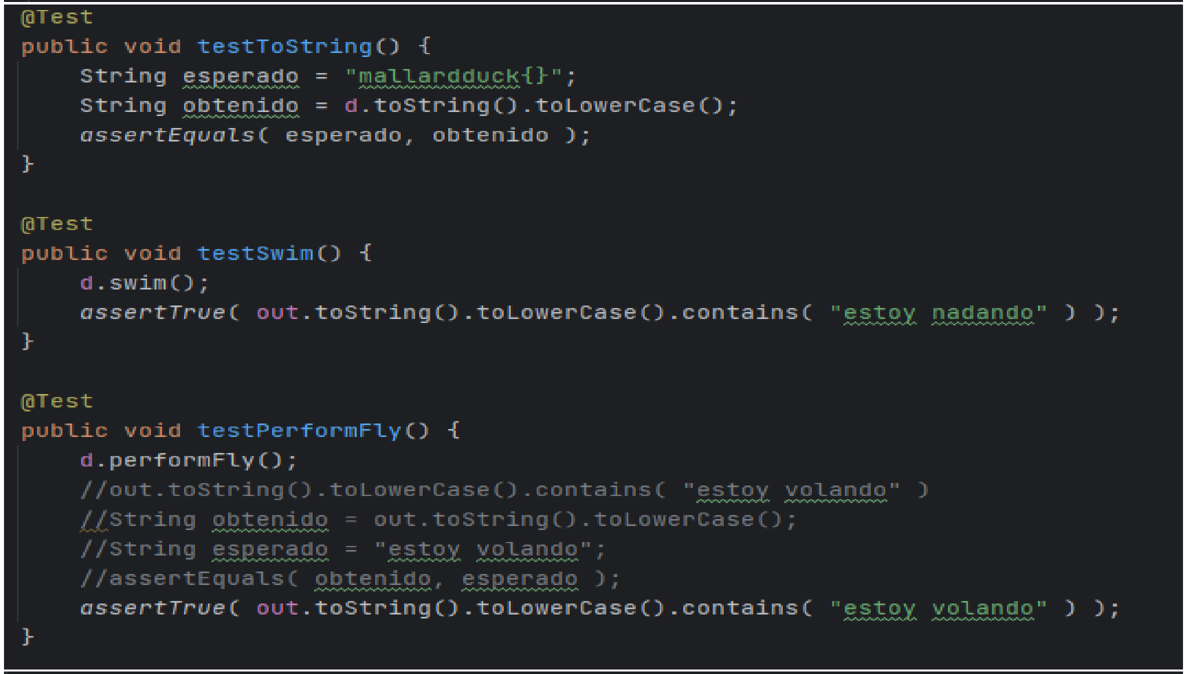


```java
    @Test
    public void testToString() {
        String esperado = "flywithwings{}";
        String obtenido = ifly.toString().toLowerCase();
        assertEquals( esperado, obtenido );
    }

    @Test
    public void fly() {
        ifly.fly();
        assertTrue( out.toString().toLowerCase().contains( "estoy volando" ) );
    }
}
```

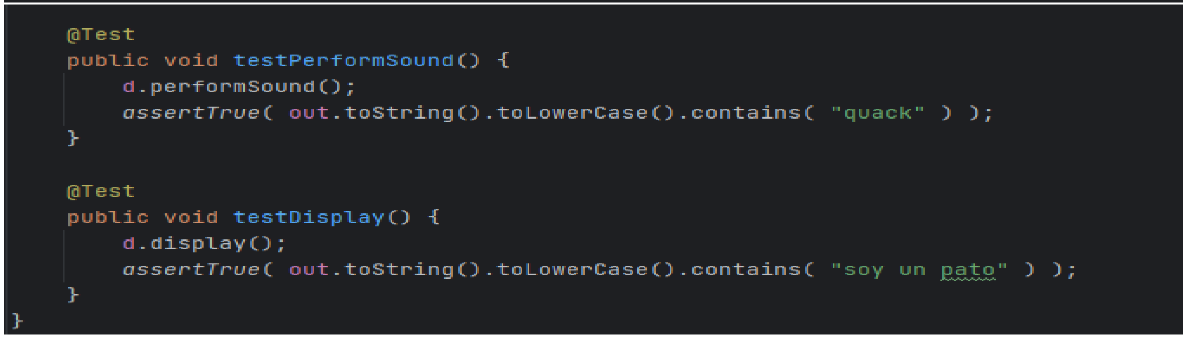
MallarDuckTest



```java
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

import java.io.ByteArrayOutputStream;
import java.io.PrintStream;

public class MallardDuckTest {
    6 usages
    private Duck d;
    6 usages
    private ByteArrayOutputStream out;

    public MallardDuckTest(){}

    @Before
    public void before() {
        d = new MallardDuck();
        out = new ByteArrayOutputStream();
        System.setOut( new PrintStream( out ) );
    }
```

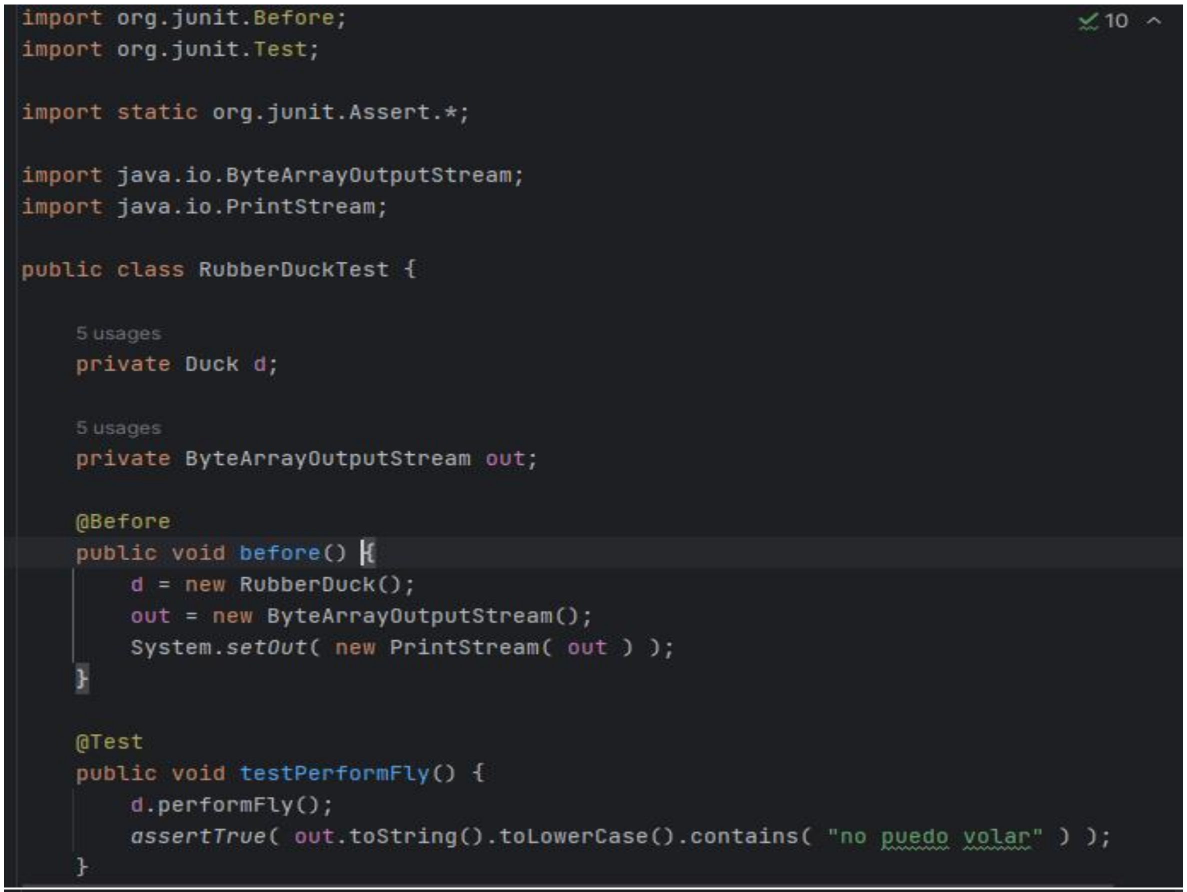


```java
@Test
public void testToString() {
    String esperado = "mallardduck{}";
    String obtenido = d.toString().toLowerCase();
    assertEquals( esperado, obtenido );
}

@Test
public void testSwim() {
    d.swim();
    assertTrue( out.toString().toLowerCase().contains( "estoy nadando" ) );
}

@Test
public void testPerformFly() {
    d.performFly();
    //out.toString().toLowerCase().contains( "estoy volando" )
    //String obtenido = out.toString().toLowerCase();
    //String esperado = "estoy volando";
    //assertEquals( obtenido, esperado );
    assertTrue( out.toString().toLowerCase().contains( "estoy volando" ) );
}
```

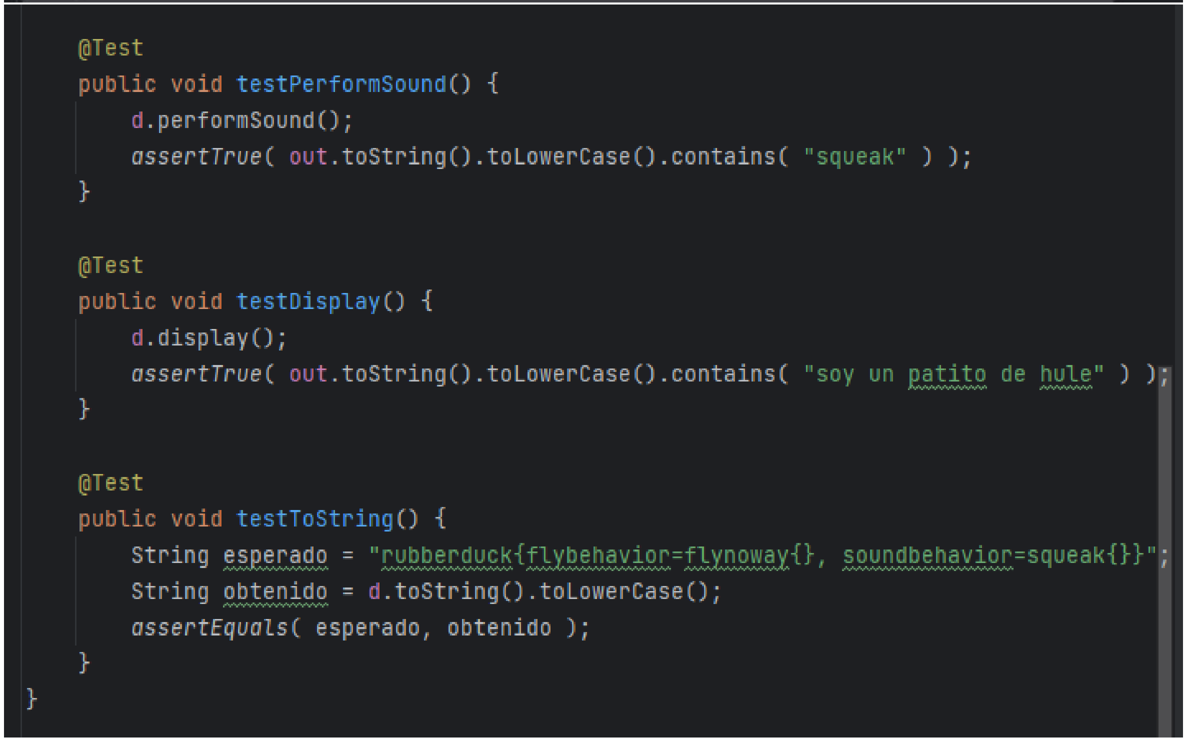


```java
    @Test
    public void testPerformSound() {
        d.performSound();
        assertTrue( out.toString().toLowerCase().contains( "quack" ) );
    }

    @Test
    public void testDisplay() {
        d.display();
        assertTrue( out.toString().toLowerCase().contains( "soy un pato" ) );
    }
}
```

RubberDuckTest



```
import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

import java.io.ByteArrayOutputStream;
import java.io.PrintStream;

public class RubberDuckTest {

    5 usages
    private Duck d;

    5 usages
    private ByteArrayOutputStream out;

    @Before
    public void before() {
        d = new RubberDuck();
        out = new ByteArrayOutputStream();
        System.setOut( new PrintStream( out ) );
    }

    @Test
    public void testPerformFly() {
        d.performFly();
        assertTrue( out.toString().toLowerCase().contains( "no puedo volar" ) );
    }
```



```
    @Test
    public void testPerformSound() {
        d.performSound();
        assertTrue( out.toString().toLowerCase().contains( "squeak" ) );
    }

    @Test
    public void testDisplay() {
        d.display();
        assertTrue( out.toString().toLowerCase().contains( "soy un patito de hule" ) );
    }

    @Test
    public void testToString() {
        String esperado = "rubberduck{flybehavior=flynoway{}, soundbehavior=squeak{}}";
        String obtenido = d.toString().toLowerCase();
        assertEquals( esperado, obtenido );
    }
}
```