```java
package com.ffcg;

import org.junit.Test;

public class DivisorTest {

    @Test
    public void return_false_for_0_as_divisor_of_any_number(){
        assert(!(new Divisor(0)).isDivisorOf(0));
        assert(!(new Divisor(0)).isDivisorOf(1));
        assert(!(new Divisor(0)).isDivisorOf(2));
        assert(!(new Divisor(0).isDivisorOf(3)));
        assert(!(new Divisor(0)).isDivisorOf(4));
    }

    @Test
    public void return_false_for_2_as_divisor_of_0() {
        assert(!(new Divisor(2)).isDivisorOf(0));
    }

    @Test
    public void return_false_for_2_as_divisor_of_1() {
        assert(!(new Divisor(2)).isDivisorOf(1));
    }


    @Test
    public void return_false_for_2_as_divisor_of_3() {
        assert(!(new Divisor(2)).isDivisorOf(3));
    }

    @Test
    public void return_false_for_4_as_divisor_of_91() {
        assert(!(new Divisor(4)).isDivisorOf(91));
    }

```

```
37      @Test
38      public void return_true_for_2_as_divisor_of_4() {
39          assert((new Divisor(2)).isDivisorOf(4));
40      }
41
42      @Test
43      public void return_true_for_3_as_divisor_of_21
    () {
44          assert((new Divisor(3)).isDivisorOf(21));
45      }
46
47      @Test
48      public void return_true_for_2_as_divisor_of_2() {
49          assert((new Divisor(2)).isDivisorOf(2));
50      }
51  }
```

```java
package com.ffcg;

import org.junit.Test;

import java.util.ArrayList;
import java.util.List;


public class SetDivisorsTest {
    @Test
    public void return_true_when_calculating_all_divisors_of_25() {
        List<Divisor> listDivisorsOf25 = new ArrayList<Divisor>();
        listDivisorsOf25.add(new Divisor(1));
        listDivisorsOf25.add(new Divisor(5));
        listDivisorsOf25.add(new Divisor(25));

        SetDivisors listDivisors = new SetDivisors(25);

        assert(listDivisors.isEqualTo(listDivisorsOf25));

    }

    @Test
    public void return_true_when_calculating_all_divisors_of_23() {
        List<Divisor> listDivisorsOf23 = new ArrayList<Divisor>();
        listDivisorsOf23.add(new Divisor(1));
        listDivisorsOf23.add(new Divisor(23));

        SetDivisors listDivisors = new SetDivisors(23);

        assert(listDivisors.isEqualTo(listDivisorsOf23));

    }
```

```
34
35      @Test
36      public void
    return_true_when_calculating_sum_of_squared_divisors_
    of_25() {
37          SetDivisors listDivisorsOf25 = new
    SetDivisors(25);
38
39          assert((1*1 + 5*5 + 25*25) ==
    listDivisorsOf25.getSumOfSquaredDivisors());
40      }
41
42      @Test
43      public void
    return_true_when_calculating_sum_of_squared_divisors_
    of_23(){
44          SetDivisors listDivisorsOf23 = new
    SetDivisors(23);
45          assert((1*1 + 23*23) == listDivisorsOf23.
    getSumOfSquaredDivisors());
46
47      }
48 }
49
```

```java
package com.ffcg;

import org.junit.Test;


public class PerfectSquaredTest {
    @Test
    public void return_true_for_numbers_are_perfect_squared() {
        assert(PerfectSquared.isPerfectSquared(1));
        assert(PerfectSquared.isPerfectSquared(4));
        assert(PerfectSquared.isPerfectSquared(9));
        assert(PerfectSquared.isPerfectSquared(16));
        assert(PerfectSquared.isPerfectSquared(25));
    }
    @Test
    public void return_false_for_numbers_are_not_perfect_squared() {
        assert(!PerfectSquared.isPerfectSquared(2));
        assert(!PerfectSquared.isPerfectSquared(3));
        assert(!PerfectSquared.isPerfectSquared(5));
        assert(!PerfectSquared.isPerfectSquared(6));
        assert(!PerfectSquared.isPerfectSquared(7));
    }
}
```

```java
package com.ffcg;

import org.junit.Test;

import java.util.Arrays;

public class SquaredDivisorTest {

    @Test
    public void return_true_calculating_all_squared_divisors_between_1_and_250() {
        int [] squaredDivisorsBetween1And250 = {1, 42, 246};

        int [] squaredDivisors = SquaredDivisor.searchSquaredDivisor(1, 250);

        assert(Arrays.equals(squaredDivisorsBetween1And250, squaredDivisors));

    }

    @Test
    public void return_true_calculating_all_squared_divisors_between_42_and_250() {
        int [] squaredDivisorsBetween1And250 = {42, 246};

        int [] squaredDivisors = SquaredDivisor.searchSquaredDivisor(42, 250);

        assert(Arrays.equals(squaredDivisorsBetween1And250, squaredDivisors));

    }


}

```