

СОФТВЕРСКИ КВАЛИТЕТ И ТЕСТИРАЊЕ

ДОМАШНА ЗАДАЧА 1 и 2

Фисник Лимани, 151027

1. Изворниот код на напишаната функција

```
1 public class Lab1_2 {  
2     public Boolean IsPowerOfTwo(Integer num)  
3     {  
4         if( num == 0) {  
5             return false;  
6         }  
7         while(num != 1) {  
8             if(num%2 != 0){  
9                 return false;  
10            }  
11            num = num / 2;  
12        }  
13        return true;  
14    }  
15 }
```

2. Тестови за дадена функција

- a. Прво го дефинираме почетниот setup т.е. креирање инстанца од класата Lab1_2 пред извршување на секој тест со помош на @BeforeEach анотацијата

```
Lab1_2 lab12;  
@BeforeEach  
void setUp() {  
    lab12 = new Lab1_2();  
}
```

- b. Тестираме со бројот 0

```
@Test  
public void testNumberZero(){  
    assertFalse(lab12.IsPowerOfTwo( num: 0));  
}
```

✓ Test Results	15 ms
✓ Lab1_2Test	15 ms
✓ testNumberZero()	15 ms

- c. Тестираме со бројот 1

```
@Test
public void testNumberOne(){
    assertTrue(lab12.IsPowerOfTwo( num: 1));
}
```

✓ Test Results	16 ms
✓ Lab1_2Test	16 ms
✓ testNumberOne()	16 ms

- d. Тестираме со број кој навистина е степен на 2, пр. бројот 2048

```
@Test
public void testTrueValue(){
    assertTrue(lab12.IsPowerOfTwo( num: 2048));
}
```

✓ Test Results	15 ms
✓ Lab1_2Test	15 ms
✓ testTrueValue()	15 ms

- e. Тестираме со број кој не е степен на 2, пр. бројот 5

```
@Test
public void testFalseValue(){
    assertFalse(lab12.IsPowerOfTwo( num: 5));
}
```

✓ Test Results	15 ms
✓ Lab1_2Test	15 ms
✓ testFalseValue()	15 ms

- f. Тестираме со негативен парен број, пр. -10

```
@Test
public void testEvenNegativeNumber(){
    assertFalse(lab12.IsPowerOfTwo( num: -10));
}
```

✓ Test Results	24 ms
✓ Lab1_2Test	24 ms
✓ testEvenNegativeNur	24 ms

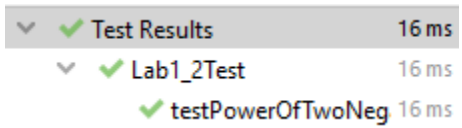
- g. Тестираме со негативен непарен број, пр. -5

```
@Test
public void testOddNegativeNumber(){
    assertFalse(lab12.IsPowerOfTwo( num: -5));
}
```

✓ Test Results	16 ms
✓ Lab1_2Test	16 ms
✓ testOddNegativeNur	16 ms

- h. Тестираме со негативен број, чиј апсолутна вредност е степен на 2, пр. -16

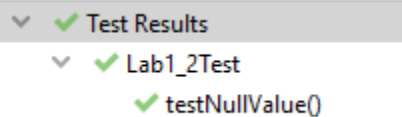
```
@Test
public void testPowerOfTwoNegativeNumber(){
    assertFalse(lab12.IsPowerOfTwo( num: -16));
}
```



Test Results 16 ms
Lab1_2Test 16 ms
testPowerOfTwoNeg 16 ms

- i. Тестирање со null вредност

```
@Test
public void testNullValue(){
    assertThrows(NullPointerException.class, () -> lab12.IsPowerOfTwo( num: null));
}
```



Test Results
Lab1_2Test
testNullValue()

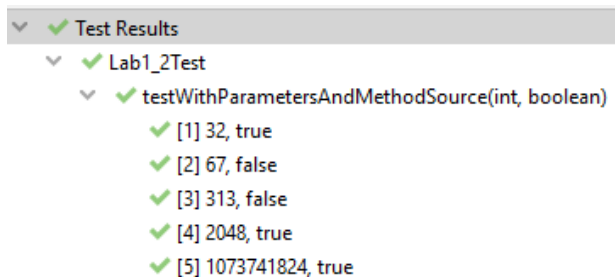
Ќе го искористиме NullPointerException што се фрли од Java.

Со овие тестови се покриваат сите можни недостатоци и испади на функцијата.

- j. Параметризирани тестови

```
@ParameterizedTest
@MethodSource("powerOfTwoNumbers")
public void testWithParametersAndMethodSource(int inputNumber, boolean expectedResult){
    assertEquals(expectedResult, lab12.IsPowerOfTwo(inputNumber));
}

// PARAMETERS
public static Collection<Object[]> powerOfTwoNumbers() {
    return Arrays.asList(new Object[][]{
        {32, true},
        {67, false},
        {313, false},
        {2048, true},
        {1073741824, true}
    });
}
```



Test Results
Lab1_2Test
testWithParametersAndMethodSource(int, boolean)
[1] 32, true
[2] 67, false
[3] 313, false
[4] 2048, true
[5] 1073741824, true

3. Ако направиме мало рефакторирање на кодот, така да ние да провериме на почеток дали вредноста е null и да фрлиме исклучок, кодот ќе изгледа вака:

```
1 public class Lab1_2 {
2     public Boolean IsPowerOfTwo(Integer num)
3     {
4         if(num == null){
5             throw new NullPointerException();
6         }
7
8         if( num == 0) {
9             return false;
10        }
11        while(num != 1) {
12            if(num%2 != 0){
13                return false;
14            }
15            num = num / 2;
16        }
17        return true;
18    }
19 }
```

Ги извршуваме тестовите, и добиваме:

Test Results	43 ms
Lab1_2Test	43 ms
testNumberOne()	12 ms
testFalseValue()	1 ms
testNumberZero()	1 ms
testWithParametersAndMethodSource(int, boolean)	24 ms
[1] 32, true	21 ms
[2] 67, false	
[3] 313, false	1 ms
[4] 2048, true	1 ms
[5] 1073741824, true	1 ms
testOddNegativeNumber()	1 ms
testEvenNegativeNumber()	1 ms
testPowerOfTwoNegativeLargeNumber()	1 ms
testPowerOfTwoNegativeNumber()	
testNullValue()	2 ms
testTrueValue()	