Fibonacci Kata

Fibonacci Sequence.

The Fibonacci Sequence is the series of numbers:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
The next number is found by adding up the two numbers before it.
The 2 is found by adding the two numbers before it (1+1)
The 3 is found by adding the two numbers before it (1+2),
And the 5 is (2+3),
and so on!

Example: the next number in the sequence above is 21+34 = **55**
It is that simple!
Here is a longer list:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, ...

The Requirements.

- Write a class named “Fibonacci” that has one method
  - generate(length: int) is called to generate the sequence of fibonacci numbers variable in length.

Begin.

- Create a project named Fibonacci
- Create a unit test named FibonacciShouldReturn

```
public class FibonacciShouldReturn {
}
```

Begin.

- Create a project named Fibonacci
- Create a unit test named FibonacciShouldReturn

```
public class FibonacciShouldReturn {
}
```

Execute this program and verify that you get the following error:

```
java.lang.Exception: No runnable methods
```

The first test.

```
import org.junit.Test;

public class FibonacciShouldReturn{

    @Test public void something(){
        Fibonacci fibonacci = new Fibonacci();
  }
}
```

The first test.

```
import org.junit.Test;

public class FibonacciShouldReturn{

    @Test public void something(){
        Fibonacci fibonacci = new Fibonacci();
  }
}
```

```
public class Fibonacci{

}
```

The first test.

```java
import org.junit.Test;

public class FibonacciShouldReturn{

    @Test public void something(){
        Fibonacci fibonacci = new Fibonacci();
  }
}
```

```java
public class Fibonacci{

}
```

The first test.

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

  @Test
    public void zeroForFirstNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(0, fibonacci.generate(1)[0]);
    }
}
```

```
public class Fibonacci{

}
```

The first test.

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

  @Test
    public void zeroForFirstNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(0, fibonacci.generate(1)[0]);
    }
}
```

```
public class Fibonacci{

    public int[] generate(int length){
        return null;
  }
}
```

The first test.

```java
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

  @Test
    public void zeroForFirstNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(0, fibonacci.generate(1)[0]);
    }
}
```

```java
public class Fibonacci{

    public int[] generate(int length){
        return null;
  }
}
```

```
java.lang.NullPointerException
```

The first test.

```java
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

  @Test
    public void zeroForFirstNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(0, fibonacci.generate(1)[0]);
    }
}
```

```
expected:<-1> but is:<0>
```

```java
public class Fibonacci{

    public int[] generate(int length){
        return new int[]{-1};
    }
}
```

The first test.

```java
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

  @Test
    public void zeroForFirstNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(0, fibonacci.generate(1)[0]);
    }
}
```

```java
public class Fibonacci{

    public int[] generate(int length){
        return new int[]{0};
    }
}
```

- Fibonacci creation is duplicated
- Generate is duplicated

The second test.

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

  @Test
    public void zeroForFirstNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(0, fibonacci.generate(1)[0]);
    }

  @Test
    public void oneForSecondNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(1, fibonacci.generate(2)[1]);
    }
}
```

```
public class Fibonacci{

    public int[] generate(int length){
        return new int[]{0};
    }
}
```

- Fibonacci creation is duplicated
- Generate is duplicated

The second test.

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

  @Test
    public void zeroForFirstNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(0, fibonacci.generate(1)[0]);
    }

  @Test
    public void oneForSecondNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(1, fibonacci.generate(2)[1]);
    }
}
```

```
public class Fibonacci{

    public int[] generate(int length){
        return new int[]{0};
    }
}
```

```
java.lang.ArrayIndexOutOfBoundsException: 1
```

- Fibonacci creation is duplicated
- Generate is duplicated

The second test.

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

  @Test
    public void zeroForFirstNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(0, fibonacci.generate(1)[0]);
    }

  @Test
    public void oneForSecondNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(1, fibonacci.generate(2)[1]);
    }
}
```

```
public class Fibonacci{

    public int[] generate(int length){
        return new int[]{0,0};
    }
}
```

```
expected:<1> but is:<0>
```

- Fibonacci creation is duplicated
- Generate is duplicated

The second test.

```java
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

  @Test
    public void zeroForFirstNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(0, fibonacci.generate(1)[0]);
    }

    @Test
    public void oneForSecondNumber(){
        Fibonacci fibonacci = new Fibonacci();
        assertEquals(1, fibonacci.generate(2)[1]);
    }
}
```

```java
public class Fibonacci{

    public int[] generate(int length){
        return new int[]{0,1};
    }
}
```

- Generate is duplicated

The second test.

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

    private Fibonacci fibonacci;

    @Before public void setUp(){
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber(){
        assertEquals(0, fibonacci.generate(1)[0]);
  }

    @Test public void oneForSecondNumber(){
        assertEquals(1, fibonacci.generate(2)[1]);
  }
}
```

```
public class Fibonacci{

    public int[] generate(int length){
        return new int[]{0,1};
    }
}
```

- Generate is duplicated

The second test.

```java
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn{

    private Fibonacci fibonacci;

    @Before public void setUp(){
        fibonacci = new Fibonacci();
    }

    @Test public void zeroForFirstNumber(){
        assertEquals(0, fibonacci.generate(1)[0]);
  }

    @Test public void oneForSecondNumber(){
        assertEquals(1, fibonacci.generate(2)[1]);
    }
}
```

```java
public class Fibonacci{

    public int[] generate(int length){
        return new int[]{0,1};
    }
}
```

The solution is not more general than the first one.
Need to identify the pattern for the numbers.

https://8thlight.com/blog/uncle-bob/2013/05/27/TheTransformationPriorityPremise.html

- Generate is duplicated

The second test.

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, fibonacci.generate(1)[0]);
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, fibonacci.generate(2)[1]);
  }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            sequence[i] = i;
        }
        return sequence;
    }
}
```

The second test.

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

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
  }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            sequence[i] = i;
        }
        return sequence;
    }
}
```

The third test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            sequence[i] = i;
        }
        return sequence;
    }
}
```

The third test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            sequence[i] = i;
        }
        return sequence;
    }
}
```

```
expected:<1> but is:<2>
```

The third test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            if (i < 2) {
                sequence[i] = i;
            } else {
                sequence[i] = 1;
            }
        }
        return sequence;
    }
}
```

The fourth test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
  }

    @Test public void twoForTheFourthNumber() {
        assertEquals(2, getNumberAtIndex(3));
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            if (i < 2) {
                sequence[i] = i;
            } else {
                sequence[i] = 1;
            }
        }
        return sequence;
    }
}
```

The fourth test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
  }

    @Test public void twoForTheFourthNumber() {
        assertEquals(2, getNumberAtIndex(3));
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            if (i < 2) {
                sequence[i] = i;
            } else {
                sequence[i] = 1;
            }
        }
        return sequence;
    }
}
```

```
expected:<2> but is:<1>
```

The fourth test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
  }

    @Test public void twoForTheFourthNumber() {
        assertEquals(2, getNumberAtIndex(3));
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            if (i < 2) {
                sequence[i] = i;
            } else {
                sequence[i] = i - 1;
            }
        }
        return sequence;
    }
}
```

- Sixth number is the next failing test

The fifth test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
  }

    @Test public void twoForTheFourthNumber() {
        assertEquals(2, getNumberAtIndex(3));
  }

    @Test public void fiveForTheFifthNumber() {
        assertEquals(5, getNumberAtIndex(5));
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            if (i < 2) {
                sequence[i] = i;
            } else {
                sequence[i] = i - 1;
            }
        }
        return sequence;
    }
}
```

The fifth test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
  }

    @Test public void twoForTheFourthNumber() {
        assertEquals(2, getNumberAtIndex(3));
  }

    @Test public void fiveForTheFifthNumber() {
        assertEquals(5, getNumberAtIndex(5));
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            if (i < 2) {
                sequence[i] = i;
            } else {
                sequence[i] = i - 1;
            }
        }
        return sequence;
    }
}
```

```
expected:<2> but is:<1>
```

The fifth test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
  }

    @Test public void twoForTheFourthNumber() {
        assertEquals(2, getNumberAtIndex(3));
  }

    @Test public void fiveForTheFifthNumber() {
        assertEquals(5, getNumberAtIndex(5));
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            if (i < 2) {
                sequence[i] = i;
            } else {
                sequence[i] = sequence[i - 1] + sequence[i - 2];
            }
        }
        return sequence;
    }
}
```

The last test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
  }

    @Test public void twoForTheFourthNumber() {
        assertEquals(2, getNumberAtIndex(3));
  }

    @Test public void fiveForTheFifthNumber() {
        assertEquals(5, getNumberAtIndex(5));
  }

    @Test public void hugeValueForFiftiethNumber() {
        assertTrue(getNumberAtIndex(49) > 1000000000);
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            if (i < 2) {
                sequence[i] = i;
            } else {
                sequence[i] = sequence[i - 1] + sequence[i - 2];
            }
        }
        return sequence;
    }
}
```

The last test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

 public class FibonacciShouldReturn {

    private Fibonacci fibonacci;

    @Before public void setUp() {
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber() {
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber() {
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber() {
        assertEquals(1, getNumberAtIndex(2));
  }

    @Test public void twoForTheFourthNumber() {
        assertEquals(2, getNumberAtIndex(3));
  }

    @Test public void fiveForTheFifthNumber() {
        assertEquals(5, getNumberAtIndex(5));
  }

    @Test public void hugeValueForFiftiethNumber() {
        assertTrue(getNumberAtIndex(49) > 1000000000);
    }

    private int getNumberAtIndex(int index) {
        return fibonacci.generate(index + 1)[index];
    }
}
```

```
public class Fibonacci {

    public int[] generate(int length) {
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            if (i < 2) {
                sequence[i] = i;
            } else {
                sequence[i] = sequence[i - 1] + sequence[i - 2];
            }
        }
        return sequence;
    }
}
```

The last test.

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

public class FibonacciShouldReturn{

    private Fibonacci fibonacci;

    @Before public void setUp(){
        fibonacci = new Fibonacci();
  }

    @Test public void zeroForFirstNumber(){
        assertEquals(0, getNumberAtIndex(0));
  }

    @Test public void oneForSecondNumber(){
        assertEquals(1, getNumberAtIndex(1));
  }

    @Test public void oneForThirdNumber(){
        assertEquals(1, getNumberAtIndex(2));
  }

    @Test public void twoForTheFourthNumber(){
        assertEquals(2, getNumberAtIndex(3));
  }

    @Test public void fiveForTheFifthNumber(){
        assertEquals(5, getNumberAtIndex(5));
  }

    @Test public void hugeValueForFiftiethNumber(){
        assertTrue(getNumberAtIndex(49) > 1000000000);
  }

    private long getNumberAtIndex(int index){
        return fibonacci.generate(index + 1)[index];
  }
}
```

```
public class Fibonacci{

    public long[] generate(int length) {
        long[] sequence = new long[length];
        for (int i = 0; i < length; i++) {
            if (i < 2) {
                sequence[i] = i;
            } else {
                sequence[i] = sequence[i - 1] +
                              sequence[i - 2];
            }
        }
        return sequence;
    }
}
```

End