JUnit 4

Testes Parametrizados

https://junit.org/junit4/javadoc/4.12/org/junit/runners/Parameterized.html

Testes Parametrizados

Usando as anotações como `@RunWith(Parameterized.class)` e `@Parameters` podemos executar o mesmo teste para múltiplos datasets.

```java
@RunWith(Parameterized.class)
public class FibonacciTest {

    private int entrada;
    private int esperado;

    public FibonacciTest (int in, int esp) {
        entrada = in;
        esperado = esp;
    }

   @Parameters
   public static Object[][] data() {
        return new Object[][] {
                { 0, 0 },
                { 1, 1 },
                { 2, 1 },
                { 6, 8 } };
    }

    @Test
    public void test() {
        assertEquals(esperado, Fibonacci.compute(entrada));
    }
}
```

```java
@RunWith(Parameterized.class)
public class FibonacciTest {

    private int entrada;
    private int esperado;

    public FibonacciTest (int in, int esp) {
        entrada = in;
        esperado = esp;
    }

   @Parameters(name= "{index}: fib[{0}]={1}")
   public static Object[][] data() {
        return new Object[][] {
                { 0, 0 },
                { 1, 1 },
                { 2, 1 },
                { 6, 8 } };
    }

    @Test
    public void test() {
        assertEquals(esperado, Fibonacci.compute(entrada));
    }
}
```

```
@RunWith(Parameterized.class)
public class FibonacciTest {

    private int entrada;
    private int esperado;

    public FibonacciTest (int in, int esp) {
        entrada = in;
        esperado = esp;
    }

   @Parameters(name= "{index}: fib[{0}]={1}")
   public static Object[][] data() {
        return ne  Object[][] {

}
```

{index} - o indice do parâmetro corrente

{0} - o valor para o primeiro parâmetro

{1} - o valor para o segundo parâmetro

```
@RunWith(Parameterized.class)
public class FibonacciTest {


 @Parameter(0)
 public int entrada;

 @Parameter(1)
 public int esperado;


 @Parameters
 public static Iterable<Object[]> data() {
     return Arrays.asList(new Object[][] {
                 { 0, 0 }, { 1, 1 }, { 2, 1 },
                 { 3, 2 }, { 4, 3 }, { 5, 5 }, { 6, 8 } });
 }

@Test
 public void test() {
     assertEquals(esperado, Fibonacci.compute(entrada));
 }
}
```


Mão na Massa

Mão na Massa 4

Crie testes parametrizados para testar o método **calculaImposto** da classe **CalculoImpostoRenda**. Refatore o método se necessário.

Mão na Massa 5

Crie casos de teste parametrizados para o método **converte()** da classe Conversor Temperatura, que converte temperaturas de Celsius para Fahrenheit e vice-versa.