Testes Unitários Automatizados

Testes Unitários

- Tem objetivo de verificar o código desenvolvido;
- **Teste de unidades** representam testes realizados em métodos.
- É fortemente recomendável a utilização de uma ferramenta automatizada;
- Os métodos ágeis advogam pela utilização de testes automatizados.
- TDD (Test Driven Development) é o desenvolvimento de software orientado a testes. Mais do que simplesmente testar seu código, TDD é uma filosofia, uma cultura. Desenvolver o teste antes de implementar a funcionalidade.
- **JUnit** é um framework para escrever e repetir/executar testes.
- O JUnit é um framework popular para testes unitários em Java e é amplamente utilizado para garantir a qualidade e robustez do código.
- O JUnit atualmente está na versão 5;
- Utilizamos anotações, por exemplo, **@Test** pra definir o método de teste;

JUnit – Principais Anotações

@Test: Marca um método como um caso de teste que será executado pelo JUnit.

@BeforeEach: Executa o método antes de cada método de teste. Usado para inicializar objetos ou preparar o ambiente.

@AfterEach: Executa o método após cada método de teste. Usado para limpar ou restaurar o ambiente.

@BeforeAll: Executa o método uma vez antes de todos os testes da classe. Deve ser static.

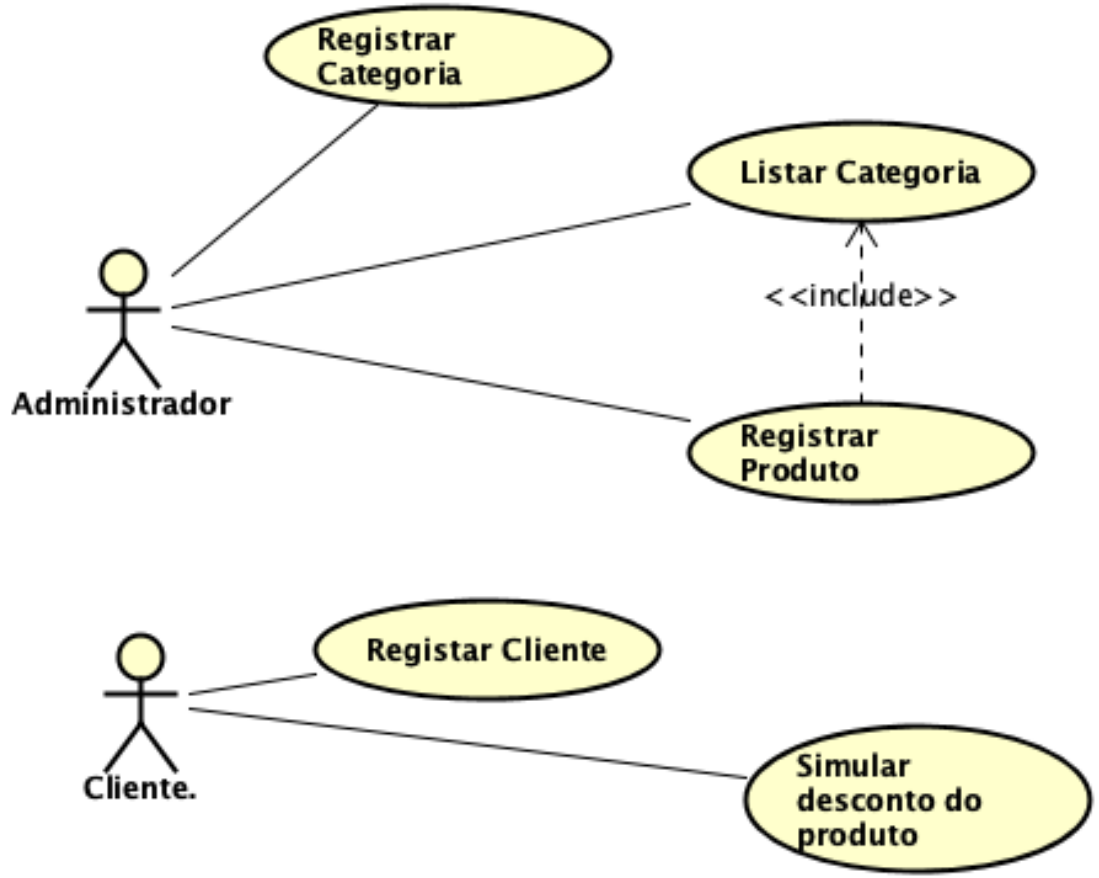
@AfterAll: Executa o método uma vez depois de todos os testes da classe. Deve ser static.

JUnit – Exemplo – Teste automatizado

```
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class CalculadoraTest {

    @Test
    void somaDeveRetornarValorCorreto() {
        Calculadora calc = new Calculadora();
        int resultado = calc.somar(2, 3);
        assertEquals(5, resultado);
    }
}
```

Projeto guiado da Disciplina (sisvenda)
Histórias de Usuário / Casos de Uso

H5. Como cliente eu quero simular um percentual de desconto de um produto para que eu possa analisar o preço final e a diferença entre os preços.



Cenários de Teste

História:

Como cliente eu quero simular um percentual de desconto de um produto para que eu possa analisar o preço final e a diferença entre os preços.

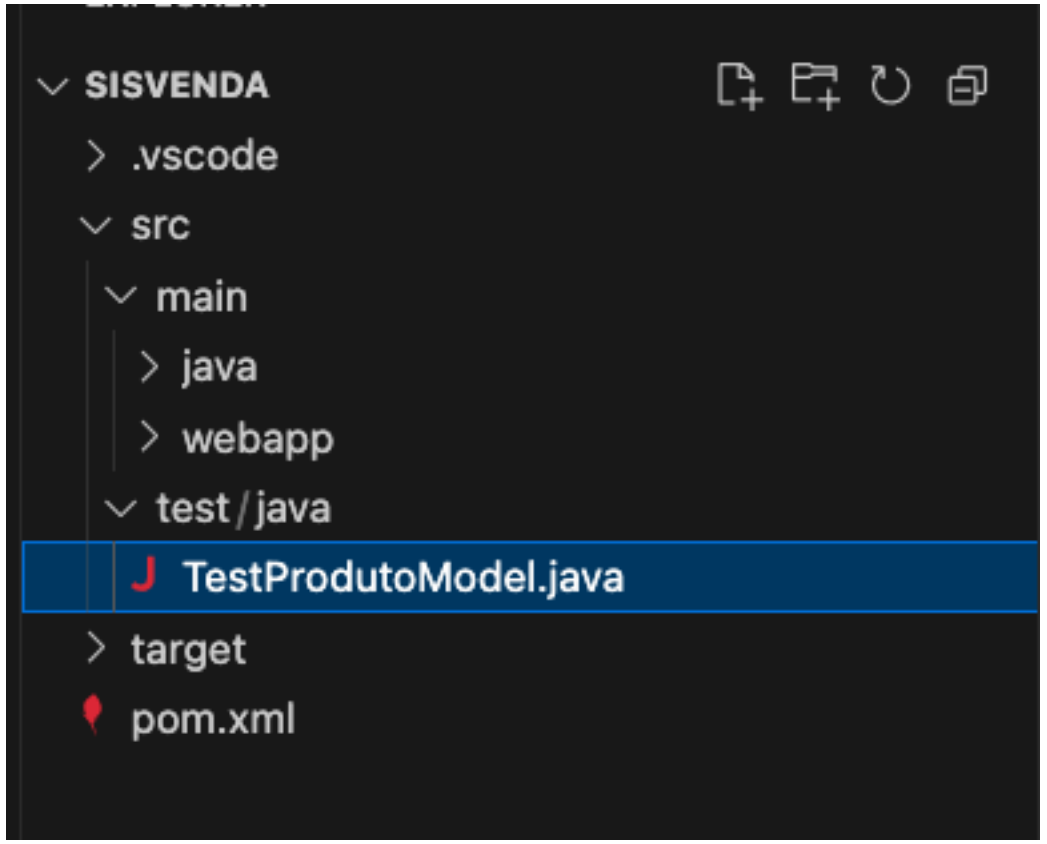
Cenário 1:
Dado que o cliente solicite uma simulação de desconto para o produto,
Quando então o cliente informar o <percentual> da simulação,
Então o sistema informará o preço do produto com desconto <resultado>

Percentual	Preço Original	Resultado
-1	1000	Valor do % deve ser positivo;
0	2350,00	Valor do % deve ser positivo
10	100,00	90,00
80	200,00	40,00

Produto (Model) – Método do desconto

```
public class Produto {

    private int id;
    private String nome;
    private float valor;
    private Categoria categoria;
    ...

public double desconto(int percentual) {
    if (percentual <= 0 || percentual > 100) {
        return 0;
    } else {
        return this.valor - this.valor * (percentual / 100.0);
        }
}
```

Criar o teste unitário (Model)



1. **Atenção: Dentro de src, crie os subdiretórios (Obrigatório):**
   - src/test/
   - src/test/java

2. **Dentro de** src/test/java, crie o arquivo TestProdutoModel.java;

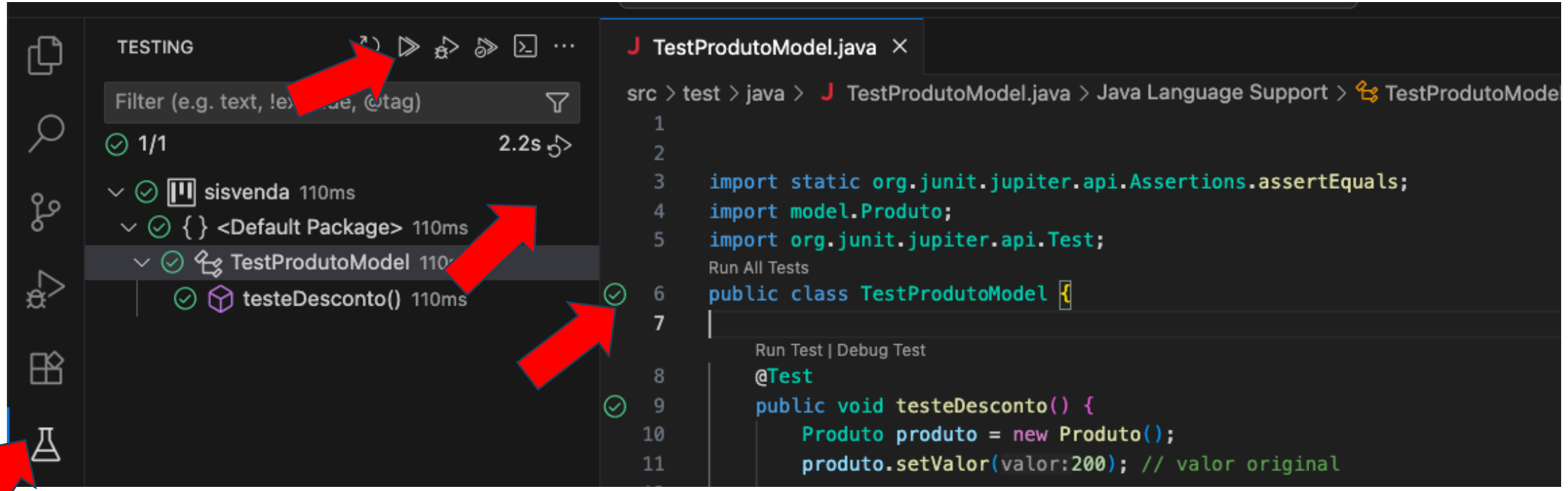
Declarar dependência do Junit 5 no pom.xml

```
<dependency>
<groupId>org.junit.jupiter</groupId>
<artifactId>junit-jupiter-api</artifactId>
<version>5.13.1</version>
<scope>test</scope>
</dependency>
```

Teste do método – Produto - ...desconto (int percentual)

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import model.Produto;
import org.junit.jupiter.api.Test;


public class TestProdutoModel {

@Test
public void testeDesconto() {
    Produto produto = new Produto();
    produto.setValor(100); // valor original

    double resultado = produto.desconto(10); // 10% de desconto
    assertEquals(90, resultado, 0.001); // Resultado 90,00
    }
}
```

Executar um teste



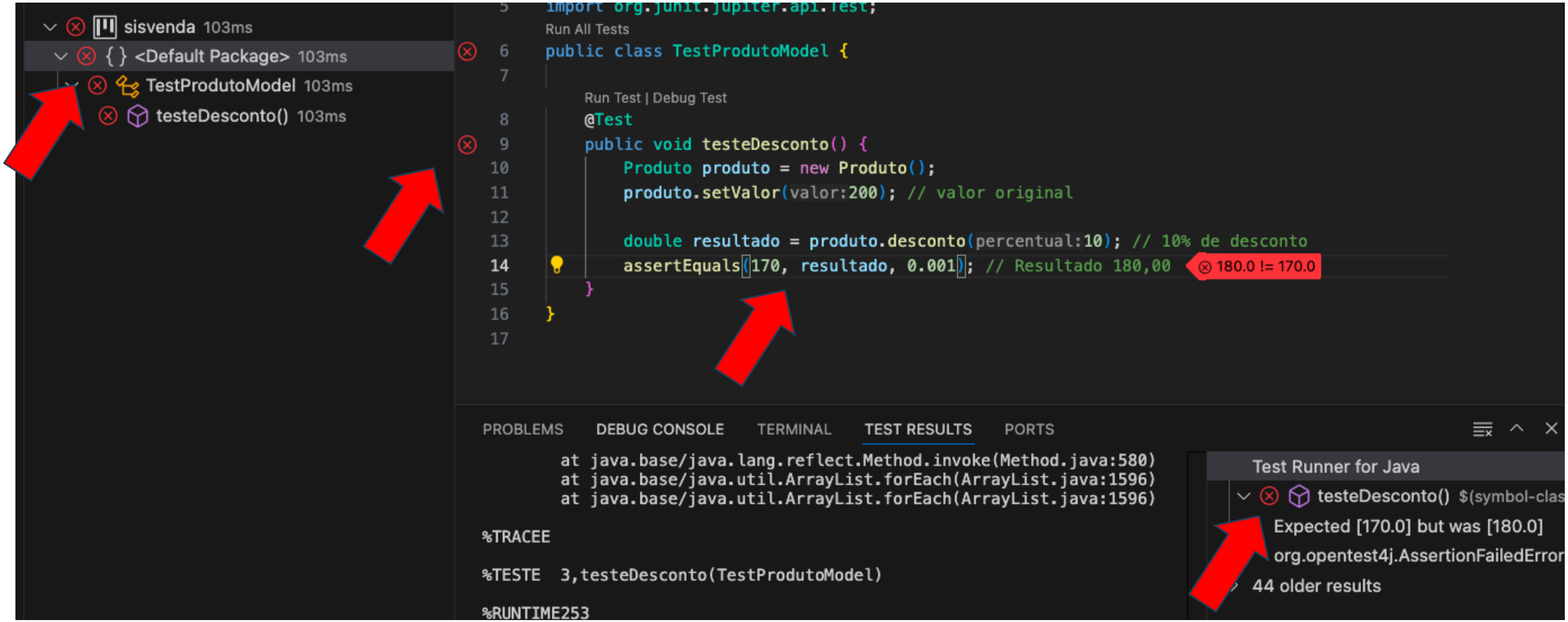
Clique nas setas.

Ícones verdes simbolizam que o teste foi executado com sucesso!!!

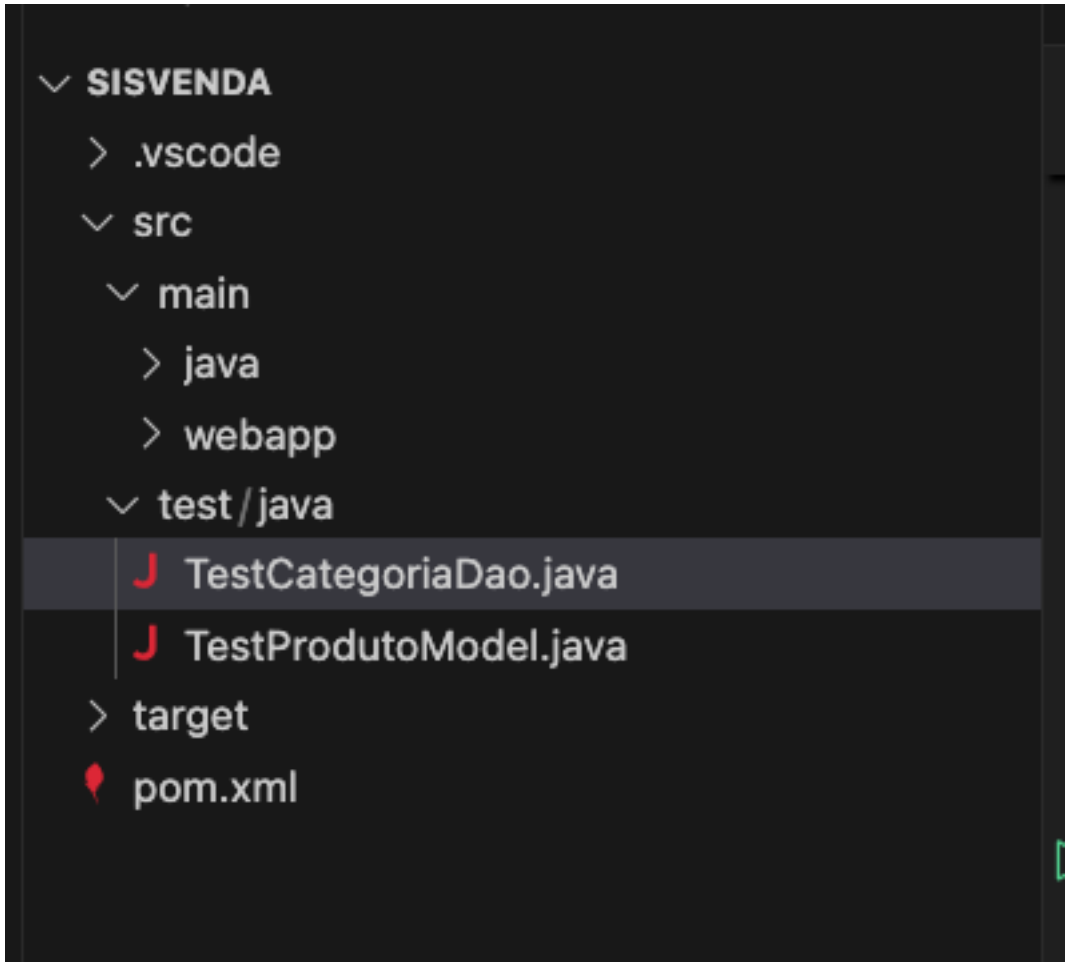
Executar o teste

Ícones verdes simbolizam que o teste foi executado com sucesso!!!

Incluindo um erro



Criar o teste unitário (CategoriaDao)



- **Dentro de** src/test/java, crie o arquivo TestCategoriaDao.java;

- **Objetivo do Teste**
  - Verificar os métodos de acesso ao banco de dados (CategoriaDao)
    - Inserir;
    - listarUm e
    - Alterar.

Categoria (Model) – Comparar Objetos

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Categoria that = (Categoria) o;
    return id == that.id && Objects.equals(nome, that.nome);
}

@Override
public int hashCode() {
    return Objects.hash(id, nome);
}
```

CategoriaDao (DAO) - Inserir

```
public int inserir2(Categoria categoria) throws SQLException {
	String sql = "INSERT INTO categoria(nome) VALUES(?)";
	PreparedStatement stmt = con.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);
	stmt.setString(1, categoria.getNome());
	stmt.executeUpdate();

	ResultSet rs = stmt.getGeneratedKeys();
	int idGerado = -1;

	if (rs.next()) {
		idGerado = rs.getInt(1);
	}
	rs.close();
	stmt.close();
	con.close();
	return idGerado;
}
```

TestCategoriaDao.java (Test)

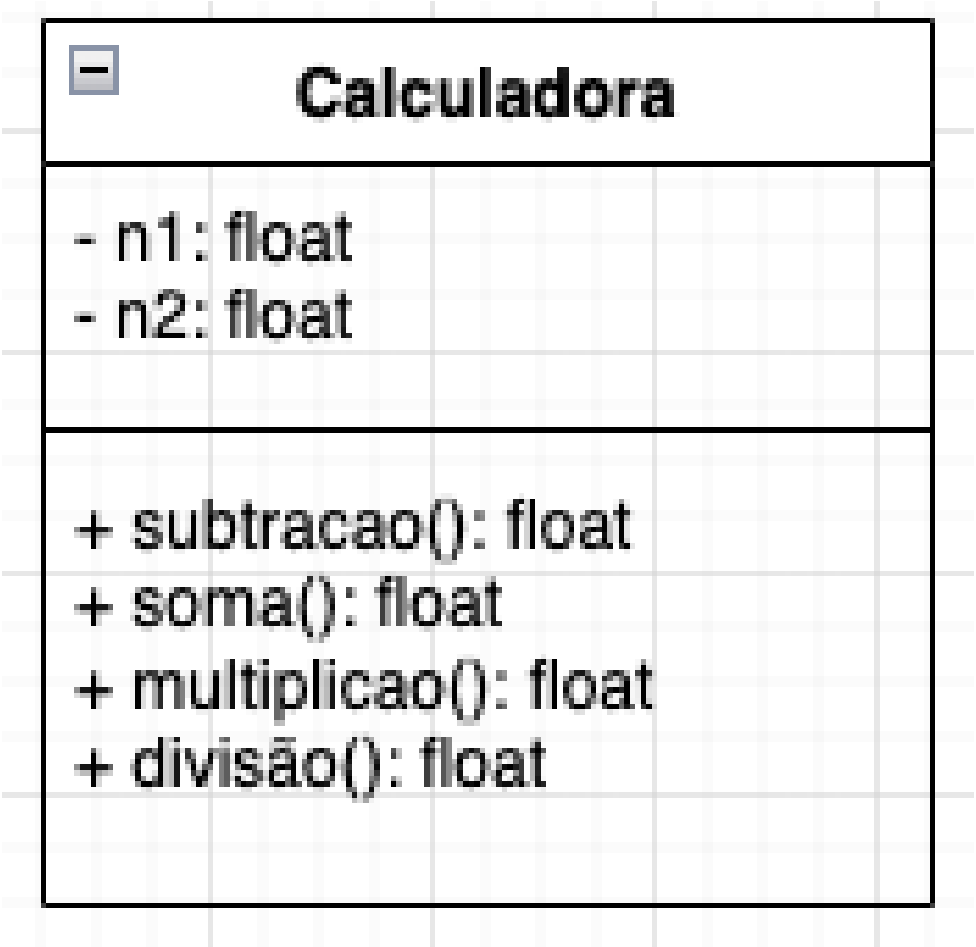
```
import dao.CategoriaDao;
import java.sql.SQLException;
import model.Categoria;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertTrue;

public class TestCategoriaDao {

    Categoria categoria;
    Categoria categoria2;
    Categoria categoria3;
    CategoriaDao categoriaDao;

@BeforeEach
public void inicializar() {
    categoria = new Categoria();
    categoria.setNome("Banana2323");
}
```

TestCategoriaDao.java (Test)

```
@Test
public void inserirBuscarAlterar() throws SQLException {
    // Inserir
        categoriaDao = new CategoriaDao();
        int id = categoriaDao.inserir2(categoria);
        categoria.setId(id);
        assertTrue(categoria.getId() > 0);

        // Buscar
        categoriaDao = new CategoriaDao();
        categoria2 = categoriaDao.listarUm(categoria.getId());
        assertEquals(categoria, categoria2);

        // Alterar
        categoriaDao = new CategoriaDao();
        categoria2.setNome("BananaNova");
        categoriaDao.alterar(categoria2);

        // Buscar categoria alterada
        categoriaDao = new CategoriaDao();
        categoria3 = categoriaDao.listarUm(categoria2.getId());
        assertEquals(categoria2.getNome(), categoria3.getNome());
}
```

Exercício - Teste Automatizado

Implemente a classe e os testes automatizados para os métodos.



Projeto (PRJ3 – Testes)

Implemente dois testes unitários em seu projeto da disciplina.

1. Teste o método com o cálculo numérico de uma classe Java Bean (model) e
2. Teste os métodos incluir, alterar, listarUm e excluir de uma classe Dao.