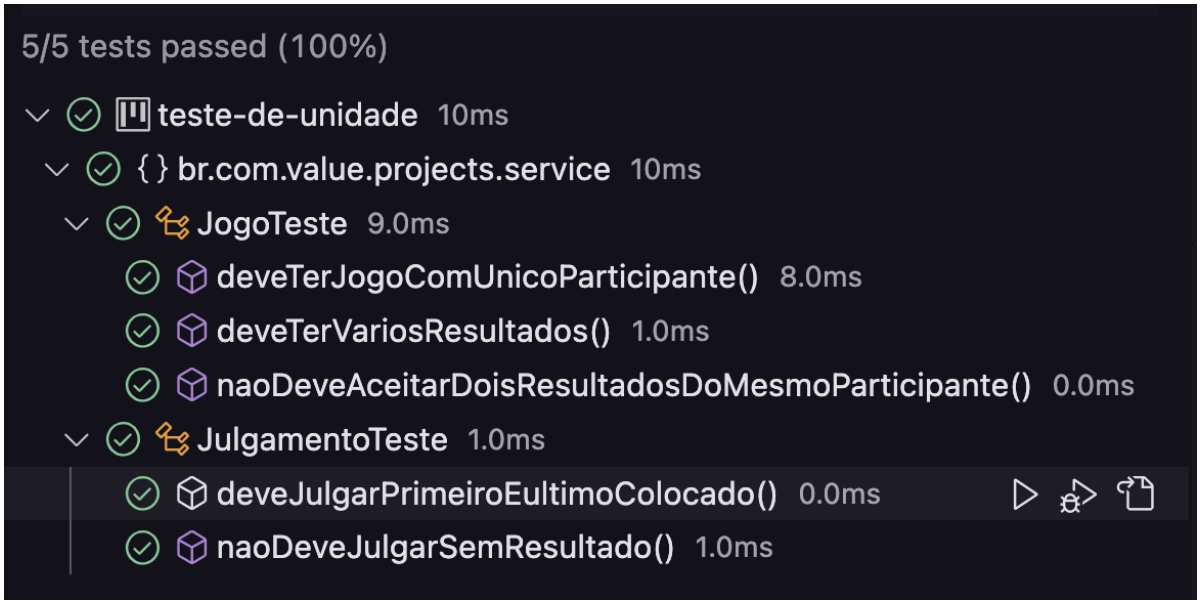
Tarefa 1 (AC1) - Ambiente de Testes com JUNIT - Rodar os testes unitários do Projeto

> Daniel De Lima Ferreira, RA: 200010
> Eduardo de Oliveira Almeida, RA: 190963
> Enrico Venturini da Costa, RA: 210XXX
> Jhonatan José Gomes do Amaral, RA: 190932

- Repositório: https://github.com/DuduAlmeida/facens.eng.softwareii/tree/main/teste-de-unidade-02-03
- Resultado dos testes:



- **JogoTeste.java**

```
package br.com.value.projects.service;

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

```
import org.junit.Test;

import br.com.value.projects.builder.CriadorDeJogo;
import br.com.value.projects.dominio.Jogo;
import br.com.value.projects.dominio.Participante;
import br.com.value.projects.dominio.Resultado;

//Aqui deixa claro que os testes serão focados para a classe de jogo
public class JogoTeste {

  @Test
  // O nome condiz com o que é testado no método
  public void deveTerJogoComUnicoParticipante() {
    // DANIEL: CENÁRIO
    // Instanciação da classe jogo
    Jogo jogo = new Jogo("Jogo de corrida");

    // DANIEL: RESULTADOS
    // Validando se o tamanho dos resultados é vazio, já que não foi adicionado
    assertEquals(0, jogo.getResultados().size());

    // DANIEL: EXECUÇÃO
    // Adicionando resultados no jogo, e um participante
    jogo.anota(new Resultado(new Participante("Leonardo"), 150));

    // DANIEL: RESULTADOS
    // Validando se o tamanho da lista de resultados é igual a 1
    assertEquals(1, jogo.getResultados().size());

    // Valida se a métrica do primeiro resultado é igual a 150 +- 0.00001
    assertEquals(150.0, jogo.getResultados().get(0).getMetrica(), 0.00001);
  }

  @Test
  // O nome condiz com o que é testado no método
  public void deveTerVariosResultados() {

    // ENRICO: CENÁRIO
    // Instanciação da classe jogo, com 2 resultados, tendo um participante em cada
    // um deles
    Jogo jogo = new CriadorDeJogo()
        .para("Cata moedas")

        // ENRICO: EXECUÇÃO
        .resultado(new Participante("Nelson"), 150.0)
        .resultado(new Participante("Pedro"), 200.0)
        .constroi();

    // EDUARDO: RESULTADOS
    // Valida se há 2 resultados no jogo
    assertEquals(2, jogo.getResultados().size());
    // Valida se a métrica do primeiro resultado é igual a 150 +- 0.00001
    assertEquals(150.0, jogo.getResultados().get(0).getMetrica(), 0.00001);
    // Valida se a métrica do segundo resultado é igual a 200 +- 0.00001
    assertEquals(200.0, jogo.getResultados().get(1).getMetrica(), 0.00001);
  }
```

```
  @Test
  // Nome condiz com a implementação do teste
  public void naoDeveAceitarDoisResultadosDoMesmoParticipante() {
    // EDUARDO: CENÁRIO

    // Instanciação de um jogo e participante
    Jogo jogo = new Jogo("Ca�a pe�as");
    Participante leonardo = new Participante("Leonardo");

    // EDUARDO: EXECUÇÃO
    // O resultado que deve ser considerado:
    jogo.anota(new Resultado(leonardo, 500.0));
    // deve ser ignorado
    jogo.anota(new Resultado(leonardo, 600.0));

    // EDUARDO: RESULTADO
    // Valida se há 1 resultado no jogo
    assertEquals(1, jogo.getResultados().size());
    // Valida se a métrica é igual ao primeiro resultado inserido (500 +- 0.00001)
    assertEquals(500, jogo.getResultados().get(0).getMetrica(), 0.00001);
  }

}
```

- **Juiz.java**

```
package br.com.value.projects.service;

import br.com.value.projects.dominio.*;

//Como os métodos dessa classe não possui @Test antes da declaração, eles não são compilados
//pelo JUnit, além disso, ao olhar a semântica da classe, verifica-se que
//o nome não demonstra que o intuito dessa classe é testar algo
//Essa classe deveria estar dentro da pasta: src/.../dominio
public class Juiz {

  private double maisPontos = Double.NEGATIVE_INFINITY;
  private double menosPontos = Double.POSITIVE_INFINITY;

  public void julga(Jogo jogo) {
    if (jogo.getResultados().size() == 0) {
      throw new RuntimeException("Sem resultados n�o h� julgamento!");
    }
    for (Resultado resultado : jogo.getResultados()) {
      if (resultado.getMetrica() > maisPontos)
        maisPontos = resultado.getMetrica();
      if (resultado.getMetrica() < menosPontos)
        menosPontos = resultado.getMetrica();

    }
  }
```

```
  public double getPrimeiroColocado() {

    return maisPontos;
  }

  public double getUltimoColocado() {

    return menosPontos;
  }

}
```

- **JulgamentoTeste.java**

```
package br.com.value.projects.service;

import org.junit.*;

import br.com.value.projects.builder.CriadorDeJogo;
import br.com.value.projects.dominio.Participante;
import br.com.value.projects.dominio.Jogo;
import br.com.value.projects.dominio.Resultado;

//Deixa claro que testará a feature de Julgamento, que não é necessariamente uma classe.
public class JulgamentoTeste {

  private Juiz juiz;
  private Participante joao;
  private Participante pedro;
  private Participante katia;
  private Participante maria;

  @Before
  // O método cria juiz é realizado antes de cada teste unitário da classe, ou
  // seja, cada método do @Test
  // Nele é instanciado alguns participantes e um juiz
  public void criaJuiz() {
    // JHONATAN: CENÁRIO

    this.juiz = new Juiz();
    this.joao = new Participante("Joao");
    this.pedro = new Participante("Pedro");
    this.katia = new Participante("Katia");
    this.maria = new Participante("Maria");

  }

  @Test
  // O nome condiz com o que é testado no método
  public void deveJulgarPrimeiroEultimoColocado() {

    // JHONATAN: CENÁRIO
    // Instanciação de um jogo
```

```
    Jogo jogo = new Jogo("Derruba barreiras");

    // JHONATAN: EXECUÇÃO
    // adicionando resultados para cada participante
    jogo.anota(new Resultado(joao, 90.0));
    jogo.anota(new Resultado(pedro, 91.0));
    jogo.anota(new Resultado(katia, 93.0));
    jogo.anota(new Resultado(maria, 94.0));

    // Pedindo para o juiz julgar o jogo
    juiz.julga(jogo);

    // Métrica do maior participante (maria)
    double vencedorJogo = 94;
    // Métrica do menor participante (joao)
    double ultimoColocadoJogo = 90;

    // JHONATAN: RESULTADOS
    // Validando se os resultados do primeiro e ultimo colocados condizem com o
    // inserido
    Assert.assertEquals(vencedorJogo, juiz.getPrimeiroColocado(), 0.00001);
    Assert.assertEquals(ultimoColocadoJogo, juiz.getUltimoColocado(), 0.00001);
  }

  // TODOS: RESULTADOS
  // Nesse arquivo não é validado com o Assert.assertEquals, o resultado é
  // validado se terá uma exceção lançada durante a execução
  @Test(expected = RuntimeException.class)
  public void naoDeveJulgarSemResultado() {
    // TODOS: CENÁRIO
    Jogo jogo = new CriadorDeJogo()
        // TODOS: EXECUÇÃO
        .para("Ca�a pe�as")
        .constroi();

    juiz.julga(jogo);
  }

}
```