UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM

INSTITUTO DE COMPUTAÇÃO- ICOMP

ESPECIALIZAÇÃO EM IA PARA ENGENHARIA DE TESTES DE SOFTWARE

Trabalho 4 - ComplexGraph + JUnit + EclEmma

Débora da Costa Medeiros

Manaus - AM

Novembro de 2023

Débora da Costa Medeiros

Trabalho 4 - ComplexGraph + JUnit + EclEmma

Trabalho submetido à avaliação, para a obtenção de nota parcial referente a discilplina de Teste, Verificação e Validação do Programa de Especialização em IA para Engenharia de Testes de Software.

Professor(a)

Raimundo Barreto, Dr.

Universidade Federal do Amazonas - UFAM

Instituto de Computação- IComp

Manaus - AM

Novembro de 2023

SUMÁRIO

1

COMPLEXGRAPH + JUNIT + ECLEMMA

O código a seguir trata da Classe AplicAprovacao, no qual serão feitos os testes.

```
package aplicacao;

public class AplicAprovacao {
    public boolean calcularAprovacao(float nota1, float nota2,
            float notafinal, int frequencia) {
        float media;
        boolean resultado;

        if (frequencia < 75){
            resultado = false;
        }
        else{
            media = (nota1 + nota2) / 2;
            if (media < 30){
                resultado = false;
            }
            else if (media >= 70){
                resultado = true;
            }
```

```
            else if ((media + notafinal)/2 >= 50){
                resultado = true;
            } else {
                resultado = false;
            }
        }

        return resultado;
    }
}
```

Utilizando a ferramenta ComplexGraph (Figura 1) foi gerado o grafo de complexidade ciclomática sobre a classe AplicAprovacao, como mostra a Figura 2.

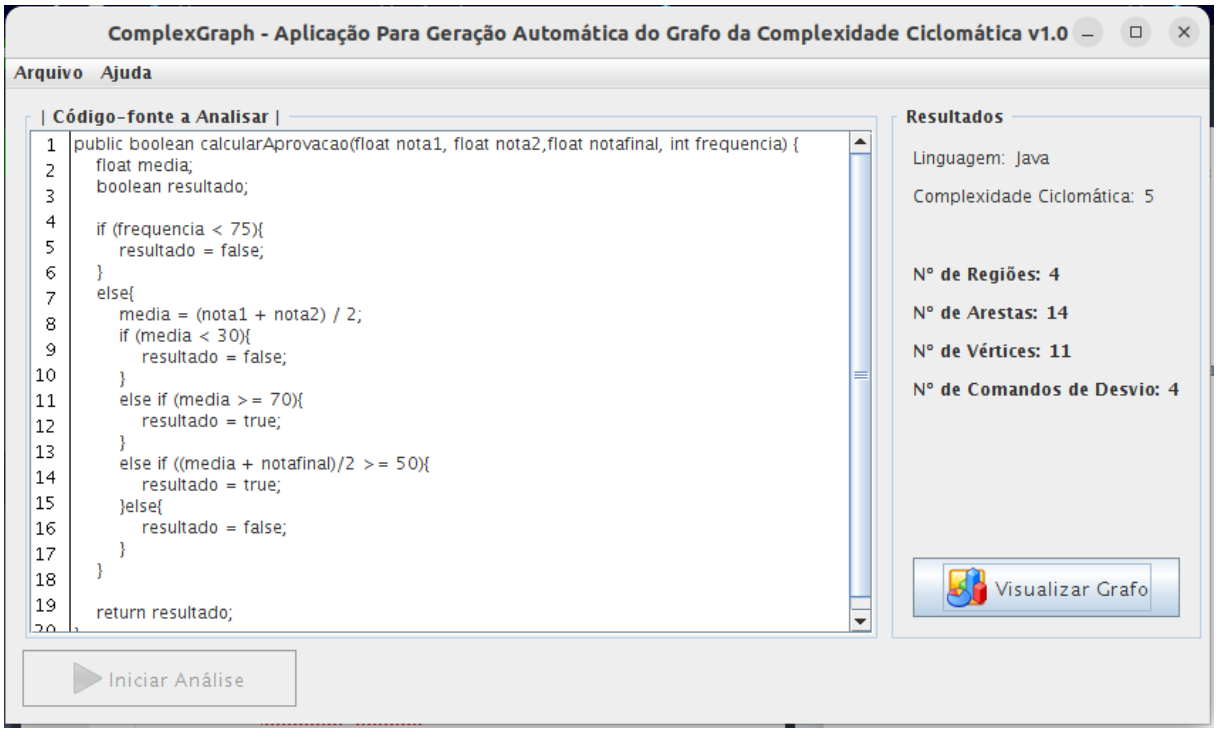

Figura 1 – ComplexGraph

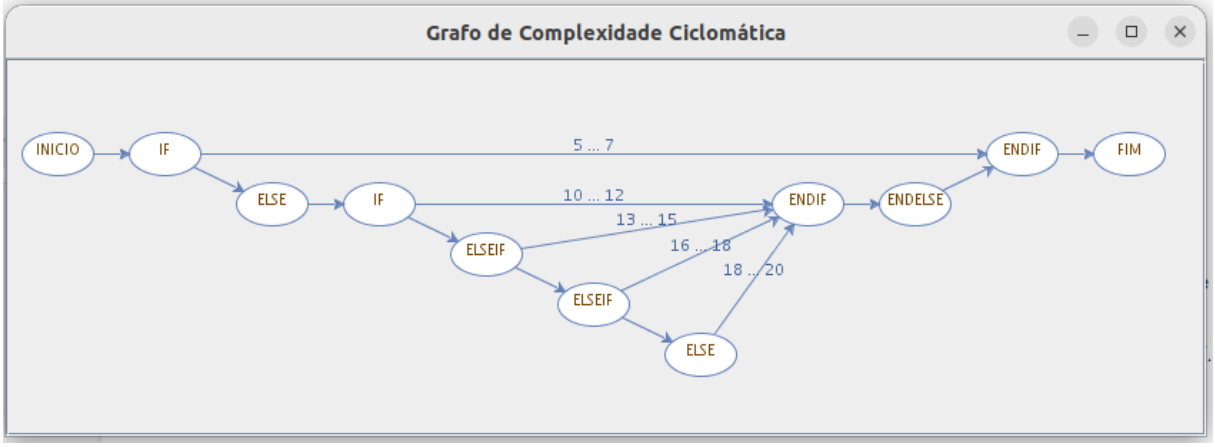

Figura 2 – Grafo de complexidade ciclomática

Foram então criados cinco casos de testes mostrados na Tabela 1.

Casos de Teste	**Entradas**	**Saída**
Caso de Teste 1	Frequência = 74	Reprovado
Caso de Teste 2	Frequência=75; Nota1=29; Nota2=30	Reprovado
Caso de Teste 3	Frequência=75; Nota1=70; Nota2=70	Aprovado
Caso de Teste 4	Frequência=75; Nota1=30; Nota2=30; NotaFinal=70	Aprovado
Caso de Teste 5	Frequência=75; Nota1=30; Nota2=30; NotaFinal=69	Reprovado

Tabela 1 – Casos de Teste

A partir dos casos de testes estabelecidos foi criado o código para classe JUnit como segue:

```
package aplicacao;

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

```
public class AplicAprovacaoTest {

    @Test
    public void testFrequenciaMenor75() {
        // Frequencia < 75
        int frequencia = 74;
        int nota1 = 0;
        int nota2 = 0;
        int notafinal = 0;
        AplicAprovacao instance = new AplicAprovacao();
        boolean expResult = false;
        boolean result = instance.calcularAprovacao(nota1, nota2,
            notafinal, frequencia);

        assertEquals(expResult, result);
    }

    @Test
    public void testMediaMenor30() {
        // Media < 30
        int frequencia = 75;
        int nota1 = 29;
        int nota2 = 30;
        int notafinal = 0;
        AplicAprovacao instance = new AplicAprovacao();
        boolean expResult = false;
        boolean result = instance.calcularAprovacao(nota1, nota2,
notafinal, frequencia);

        assertEquals(expResult, result);
```

```
    }

    @Test
    public void testMediaMaior70() {
        // Media >= 70
        int frequencia = 75;
        int nota1 = 70;
        int nota2 = 70;
        int notafinal = 0;
        AplicAprovacao instance = new AplicAprovacao();
        boolean expResult = true;
        boolean result = instance.calcularAprovacao(nota1, nota2,
notafinal, frequencia);

        assertEquals(expResult, result);
    }

    @Test
    public void testMediaFinalMaior50() {
        // (Nota Final + M dia)/ 2 >= 50
        int frequencia = 75;
        int nota1 = 30;
        int nota2 = 30;
        int notafinal = 70;
        AplicAprovacao instance = new AplicAprovacao();
        boolean expResult = true;
        boolean result = instance.calcularAprovacao(nota1, nota2,
notafinal, frequencia);

        assertEquals(expResult, result);
```

```
    }

    @Test
    public void testMediaEntre30e70eMediaFinalMenor50() {
        // Media entre 30 e 70 e M dia na Prova Final < 50
        int frequencia = 75;
        int nota1 = 30;
        int nota2 = 30;
        int notafinal = 69;
        AplicAprovacao instance = new AplicAprovacao();
        boolean expResult = false;
        boolean result = instance.calcularAprovacao(nota1, nota2,
notafinal, frequencia);

        assertEquals(expResult, result);
    }
}
```

Foram executados os testes no JUnit como mostra a Figura 3.

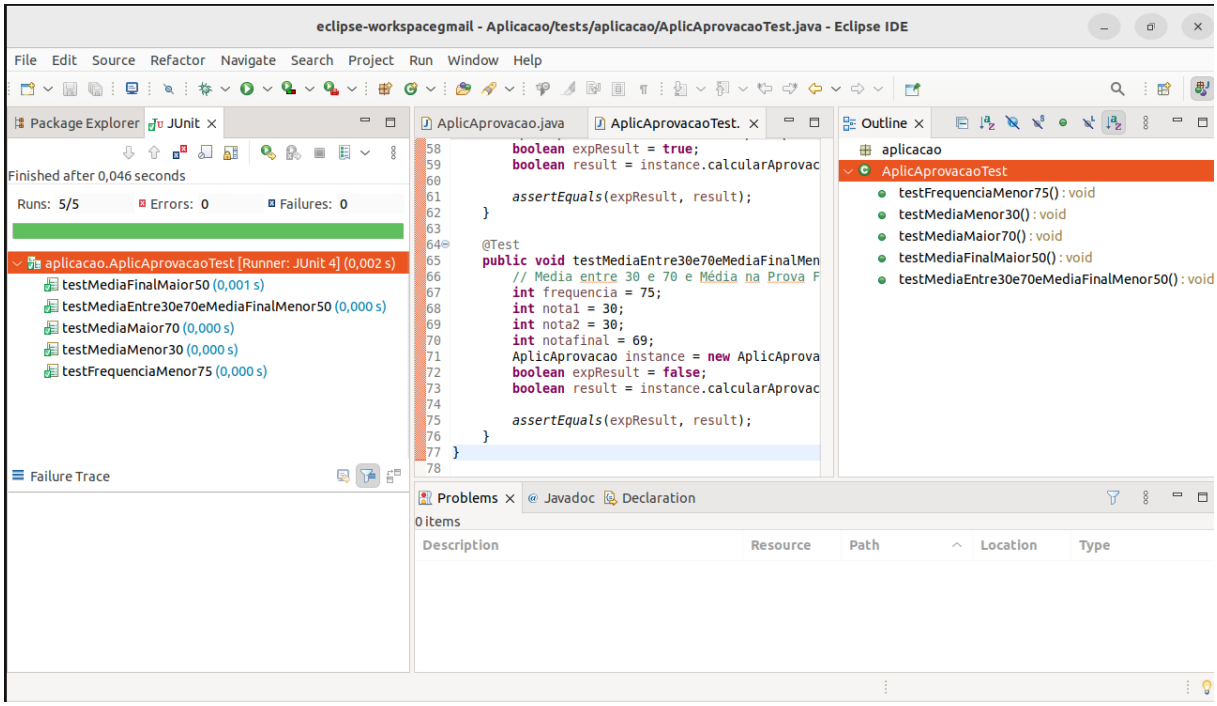

Figura 3 – Testes no JUnit

E em seguida foi aplicado o Teste de Cobertura como mostra a Figura 4, obtendo como resultado a cobertura total do código.

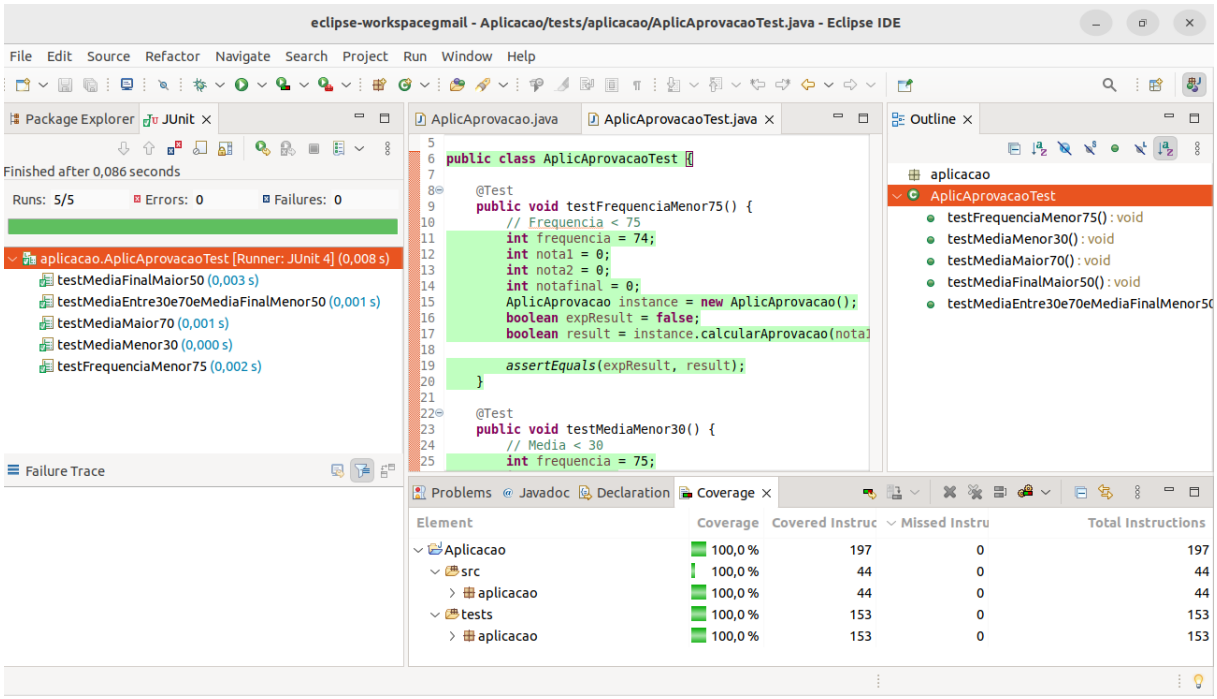

Figura 4 – Teste de Cobertura

Foram incluídos três casos de teste, mesmo após alcançar uma cobertura completa do código. A necessidade de testes de Cenários de Borda e Condições Limítrofes persiste, pois essa abordagem contribui significativamente para assegurar que o código seja capaz de lidar corretamente com todas as possíveis entradas. Desta forma é possível validar o comportamento do código em situações extremas e específicas, garantindo uma maior confiabilidade e robustez em cenários críticos. A tabela 2 mostra uma visão mais detalhada desses cenários de teste adicionais.

Casos de Teste	**Entradas**	**Saída**
Caso de Teste 6	Frequência=75; Nota1=50; Nota2=60 NotaFinal=0	Reprovado
Caso de Teste 7	Frequência=80; Nota1=40; Nota2=50 NotaFinal=0	Reprovado
Caso de Teste 8	Frequência=80; Nota1=40; Nota2=60 NotaFinal=50	Aprovado

Tabela 2 – Cenários do teste de aceitação

```
@Test
public void testFrequenciaIgual75() {
    // Frequencia = 75
    int frequencia = 75;
    int nota1 = 50;
    int nota2 = 60;
    int notafinal = 0;
    AplicAprovacao instance = new AplicAprovacao();
    boolean expResult = false;
    boolean result = instance.calcularAprovacao(nota1, nota2,
notafinal, frequencia);
```

```
        assertEquals(expResult, result);
    }

    @Test
    public void testNotaFinalIgualZero() {
        // Nota Final = 0
        int frequencia = 80;
        int nota1 = 40;
        int nota2 = 50;
        int notafinal = 0;
        AplicAprovacao instance = new AplicAprovacao();
        boolean expResult = false;
        boolean result = instance.calcularAprovacao(nota1, nota2,
notafinal, frequencia);

        assertEquals(expResult, result);
    }

    @Test
    public void testMediaEntre30e70eMediaFinalIgual50() {
        // Media entre 30 e 70 e M dia na Prova Final = 50
        int frequencia = 80;
        int nota1 = 40;
        int nota2 = 60;
        int notafinal = 50;
        AplicAprovacao instance = new AplicAprovacao();
        boolean expResult = true;
        boolean result = instance.calcularAprovacao(nota1, nota2,
notafinal, frequencia);
```

```
        assertEquals(expResult, result);
    }
```

Após a inserção dos novos casos de testes foi executados os testes no JUnit como mostra a figura 5, e foi aplicado o Teste de Cobertura como mostra a Figura 6.

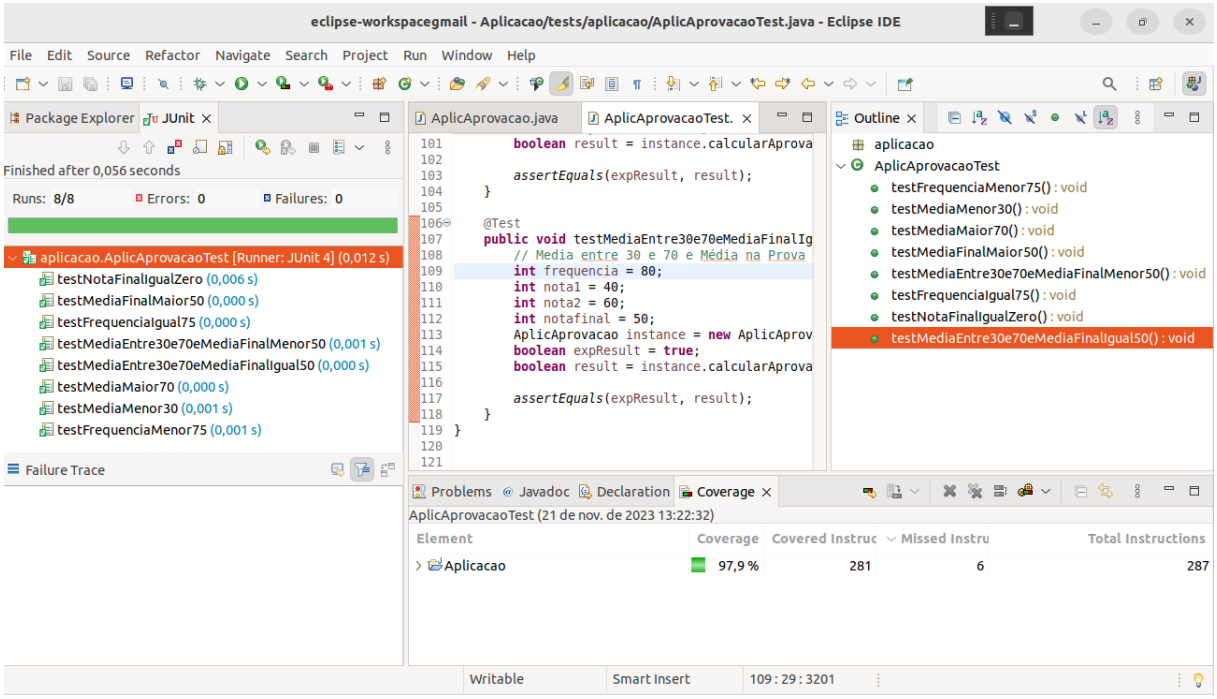

Figura 5 – Testes no JUnit

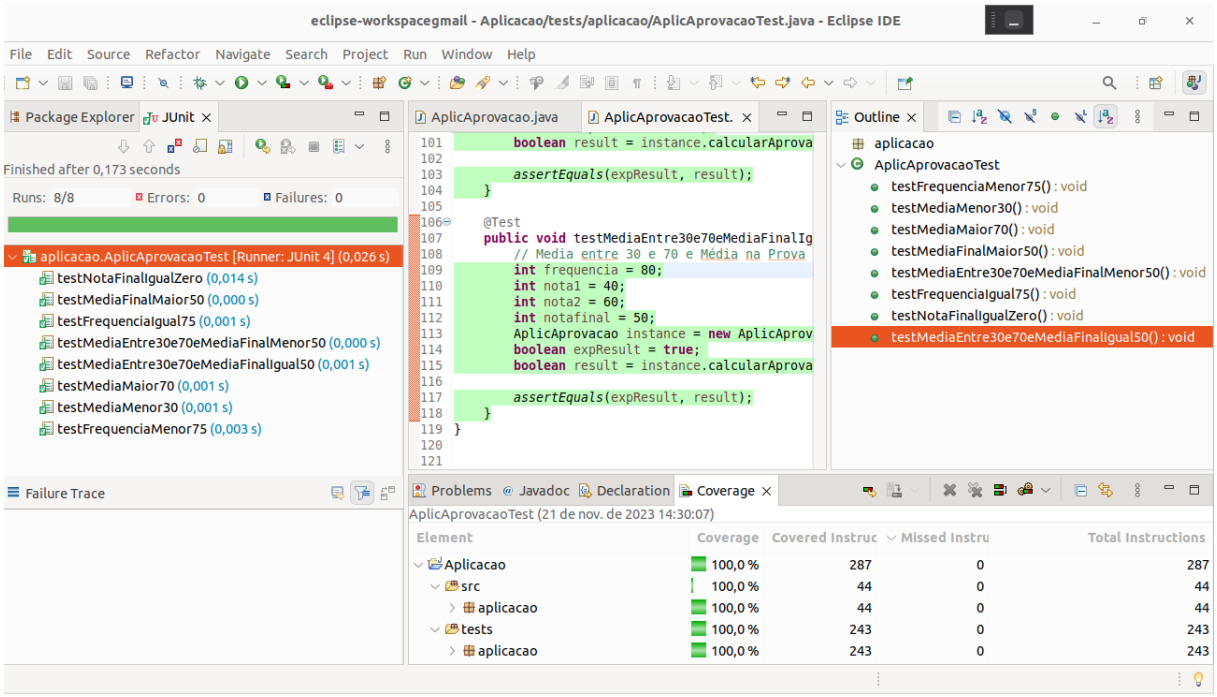

Figura 6 – Teste de Cobertura