UNIVERSIDADE SÃO JUDAS TADEU

CIÊNCIA DA COMPUTAÇÃO

GABRIEL FORNICOLA AMORIM - 824148690
GIOVANNI RIBEIRO IANNACE - 82421986
GIOVANNA FONTES DA SILVA – 823148980
LUCAS GASPARETTO NARCIZO DE MORAIS - 82426494
RAPHAEL MIGUEL FOLEGO - 822163593

Gestão e Qualidade de Software São Paulo

São Paulo

2025

Faculdade São Judas Tadeu

Exercício Prático 1 – TDD

Aplicar a técnica de TDD (Test-Driven Development) em um método como buscaBinaria (busca binária) significa seguir o ciclo:

1. Escrever o teste primeiro (teste falha porque a implementação ainda não existe ou está incorreta).
2. Implementar o código mínimo para fazer o teste passar.
3. Refatorar o código para melhorar sua estrutura, mantendo os testes verdes.

Análise do código fornecido:

Você forneceu uma implementação com erros de sintaxe e lógica. Aqui está uma versão corrigida do método buscaBinaria:

```
public static int buscaBinaria(int[] array, int valor) {
   int inicio = 0;
   int fim = array.length - 1;
   int retorno = -1;
   while (inicio <= fim) {
      int meio = (inicio + fim) / 2;
      if (array[meio] == valor) {
         retorno = meio;
         break;
      } else if (valor > array[meio]) {
         inicio = meio + 1;
      } else {
         fim = meio - 1;
      }
   }
   return retorno;
}
```

Agora, vamos aplicar TDD com teste de caixa-branca em Java usando JUnit.

Etapas do TDD com Teste de Caixa-Branca:

1. Escrever Testes (antes de implementar, no TDD clássico)

Exemplo com JUnit 5:

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

public class BuscaBinariaTest {
    @Test
    public void testValorNoMeio() {
        int[] array = {1, 3, 5, 7, 9};
        int valor = 5;
        int esperado = 2;
        assertEquals(esperado, BuscaBinaria.buscaBinaria(array, valor));
    }

    @Test
    public void testValorNoInicio() {
        int[] array = {1, 3, 5, 7, 9};
        int valor = 1;
        int esperado = 0;
        assertEquals(esperado, BuscaBinaria.buscaBinaria(array, valor));
    }

    @Test
    public void testValorNoFim() {
        int[] array = {1, 3, 5, 7, 9};
        int valor = 9;
        int esperado = 4;
```

```
        assertEquals(esperado, BuscaBinaria.buscaBinaria(array, valor));
    }

        @Test
    public void testValorNaoEncontrado() {
        int[] array = {1, 3, 5, 7, 9};
        int valor = 6;
        int esperado = -1;
        assertEquals(esperado, BuscaBinaria.buscaBinaria(array, valor));
    }

        @Test
    public void testArrayVazio() {
        int[] array = {};
        int valor = 10;
        int esperado = -1;
        assertEquals(esperado, BuscaBinaria.buscaBinaria(array, valor));
    }
}
```

Esses testes cobrem os caminhos possíveis do código (teste de caixa-branca), como:

- valor está no meio
- valor está no início
- valor está no fim
- valor não está no array
- array vazio

2. Implementar Código para Passar os Testes

3. Refatorar

Exercício Prático 2 – BDD

Vamos aplicar BDD (Behavior-Driven Development) com teste de caixa-preta ao cenário descrito: cálculo da média final dos alunos com base nas notas A1 e A2, e sua aprovação ou reprovação.

Requisito de Negócio:

"Como professor, quero saber se um aluno está aprovado ou reprovado com base na média entre as notas A1 e A2."

Especificação BDD em Gherkin:

Funcionalidade: Avaliação de aprovação dos alunos com base nas notas A1 e A2

Cenário: Aluno é aprovado com média exata de 5,0
Dado que o aluno tirou 5,0 na A1 e 5,0 na A2
Quando for calculada a média final
Então o resultado deve ser "Aprovado"

Cenário: Aluno é aprovado com média superior a 5,0
Dado que o aluno tirou 6,0 na A1 e 7,0 na A2
Quando for calculada a média final
Então o resultado deve ser "Aprovado"

Cenário: Aluno é reprovado com média inferior a 5,0
Dado que o aluno tirou 4,0 na A1 e 3,0 na A2
Quando for calculada a média final
Então o resultado deve ser "Reprovado"

Cenário: Aluno com nota fora do intervalo válido
Dado que o aluno tirou 11,0 na A1 e 5,0 na A2
Quando for calculada a média final
Então o sistema deve informar "Nota inválida"

Cenário: Aluno com nota decimal inválida
Dado que o aluno tirou 4,3 na A1 e 6,2 na A2
Quando for calculada a média final
Então o sistema deve informar "Nota inválida"

Possível Implementação Java para esse comportamento:

```
        public class Avaliacao {
    public static String avaliarAluno(double a1, double a2) {
        if (!notaValida(a1) || !notaValida(a2)) {
            return "Nota inválida";
        }
        double media = (a1 + a2) / 2;
        if (media >= 5.0) {
            return "Aprovado";
        } else {
            return "Reprovado";
        }
    }

        private static boolean notaValida(double nota) {
        return nota >= 0 && nota <= 10 && nota * 10 % 5 == 0;
    }
}
```

Classe de Step Definitions (para usar com Cucumber):

```
        import static org.junit.jupiter.api.Assertions.*;
import io.cucumber.java.pt.*;

        public class AvaliacaoSteps {
    private double a1, a2;
    private String resultado;
```

```
    @Dado("que o aluno tirou {double} na A1 e {double} na A2")
  public void queOAlunoTirouNotas(double nota1, double nota2) {
    this.a1 = nota1;
    this.a2 = nota2;
  }

    @Quando("for calculada a média final")
  public void forCalculadaAMediaFinal() {
    resultado = Avaliacao.avaliarAluno(a1, a2);
  }

    @Entao("o resultado deve ser {string}")
  public void oResultadoDeveSer(String esperado) {
    assertEquals(esperado, resultado);
  }

    @Entao("o sistema deve informar {string}")
  public void oSistemaDeveInformar(String esperado) {
    assertEquals(esperado, resultado);
  }
}
```