Universidade São Judas Tadeu - Butantã
Noturno

Nomes: **Gabriel Carvalho dos Santos - 821159957**
Gustavo Mendes de Lima - 821139542
Matheus dos Santos de Souza - 821133576
Lucas Rodrigues Santos - 823124699

Turma: GQS-CCP1AN-BUE1

Professor: Robson Calvetti

Exercício prático 1:

Fase Vermelha:

```
public class ExemploTDD {
    public static void main(String[] args) {
        OrdenaTest tdd = new OrdenaTest();
    }
}

public class OrdenaTest {
    public OrdenaTest() {
        int proposto[] = new int[] {10, 9};
        int esperado[] = new int[] {9, 10};

        Ordena teste = new Ordena();
        teste.ordenaNumerosCrescentes(proposto);

        System.out.println("Teste de Ordenação\n==========================");
        System.out.println("Ficou com o mesmo tamanho: " + caso1Test(proposto.length,
esperado.length));
        System.out.println("Ordenou com sucesso......: " + caso2Test(proposto, esperado));
    }

    public boolean caso1Test(int tamprop, int tamesp) {
        boolean resp = true;
        if (tamprop != tamesp) resp = false;
        return resp;
    }

    public boolean caso2Test(int prop[], int esp[]) {
        return numerosIguais(prop, esp);
    }

    public boolean numerosIguais(int nums1[], int nums2[]) {
        boolean resultado = true;
        for (int i = 0; i < nums1.length; i++) {
            if (nums1[i] != nums2[i]) {
                resultado = false;
                break;
            }
        }
        return resultado;
    }
}

public class Ordena {
```

```
    public void ordenaNumerosCrescentes(int iVet[]) {
        int iA, iB, iT;
        for (iA = 1; iA < iVet.length; iA++) {
            for (iB = iVet.length; iB <= iA; iB--) {
                if (iVet[iB - 1] > iVet[iB]) {
                    iT = iVet[iB - 1];
                    iVet[iB - 1] = iVet[iB];
                    iVet[iB] = iT;
                }
            }
        }
    }
}
```

```
Teste de Ordenação
==========================
Ficou com o mesmo tamanho: true
Ordenou com sucesso......: false
```

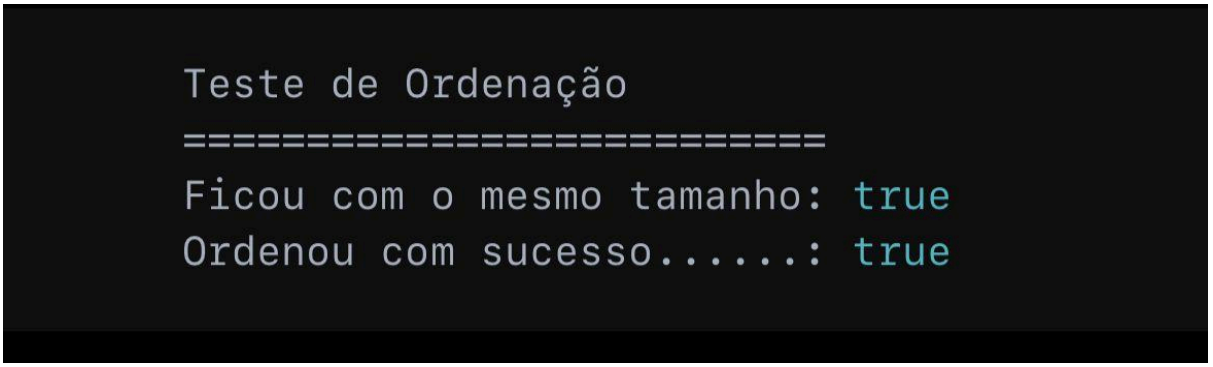
Fase verde

```
public class Ordena {
    public void ordenaNumerosCrescentes(int iVet[]) {
        // Ajuste simples para corrigir a ordenação
        for (int iA = 0; iA < iVet.length - 1; iA++) {
            for (int iB = 0; iB < iVet.length - 1 - iA; iB++) {
                if (iVet[iB] > iVet[iB + 1]) {
                    int iT = iVet[iB];
                    iVet[iB] = iVet[iB + 1];
                    iVet[iB + 1] = iT;
                }
            }
        }
    }
}
```

```
Teste de Ordenação
==========================
Ficou com o mesmo tamanho: true
Ordenou com sucesso......: true
```

Refatoração

```
public class Ordena {
    public void ordenaNumerosCrescentes(int[] iVet) {
        // Usando o método Arrays.sort para simplificar
        java.util.Arrays.sort(iVet);
    }
}
```



Exercício prático 2:

Técnica de BDD; Sistema de validação de usuário:

Funcionalidade: Sistema de validação de usuário

Cenário: Login com credenciais válidas

- Dado que o usuário possui um login "usuario_teste" e senha "senha_segura"
- Quando o usuário tenta fazer login com login "usuario_teste" e senha "senha_segura"
- Então o sistema deve validar o usuário
- E o sistema deve autorizar o acesso

Cenário: Login com credenciais inválidas

- Dado que o usuário possui um login "usuario_teste" e senha "senha_segura"
- Quando o usuário tenta fazer login com login "usuario_teste" e senha "senha_incorreta"
- Então o sistema deve invalidar o usuário
- E o sistema deve recusar o acesso

Cenário: Login com login inexistente

- Quando o usuário tenta fazer login com login "login_inexistente" e senha "qualquer_senha"
- Então o sistema deve invalidar o usuário
- E o sistema deve recusar o acesso

Cenário: Login sem fornecer credenciais

- Quando o usuário tenta fazer login sem fornecer login ou senha
- Então o sistema deve invalidar o usuário
- E o sistema deve recusar o acesso

Cenário: Login com senha em branco

- Dado que o usuário possui um login "usuario_teste" e senha "senha_segura".
- Quando o usuário tenta fazer login com login "usuario_teste" e senha em branco
- Então o sistema deve invalidar o usuário
- E o sistema deve recusar o acesso

Cenário: Login com tentativa excessiva

- Dado que o usuário tentou fazer login 5 vezes com credenciais inválidas
- Quando o usuário tenta fazer login com login "usuario_teste" e senha "senha_incorreta"
- Então o sistema deve bloquear o acesso temporariamente
- E o sistema deve exibir uma mensagem de erro informando que o acesso está bloqueado

Requisitos Funcionais:

Cadastro de Usuário

- O sistema deve permitir o cadastro de novos usuários com login e senha.
- O sistema deve validar se o login já está em uso.

Validação de Credenciais

- O sistema deve validar as credenciais (login e senha) fornecidas pelo usuário.
- O sistema deve autorizar o acesso ao sistema se as credenciais forem válidas.
- O sistema deve recusar o acesso se as credenciais forem inválidas.

Gerenciamento de Tentativas de Login

- O sistema deve contar o número de tentativas de login inválidas.
- Após um número definido de tentativas inválidas, o sistema deve bloquear temporariamente o acesso do usuário.

Recuperação de Senha

- O sistema deve permitir que usuários solicitem a recuperação de senha.
- O sistema deve enviar um e-mail de recuperação com instruções para redefinir a senha.

Requisitos Não Funcionais:

Segurança

- Armazenamento seguro de senhas.
- Proteção contra ataques de força bruta (limitação de tentativas de login).

Usabilidade

- Interface clara e amigável.
- Mensagens de erro informativas.

Desempenho

- Tempo de resposta de login inferior a 2 segundos.
- Suporte a múltiplas tentativas de login simultâneas (ex: 1000 usuários).