UNIVERSIDADE SÃO JUDAS TADEU

-

CAMPUS BUTANTÃ

GESTÃO E QUALIDADE DE SOFTWARE – CCP1AN-BUE1

Nome do Grupo

G6-GQS-CCP1AN-BUE1

Membro(s) do Grupo

822160071 – FABRÍCIO PERES – CCP - 822160071@ulife.com.br

824116869 – HERMANO PEREIRA DE SOUSA – CCP -824116869@ulife.com.br

822127136 – JONATA PABLO GARCIA – CCP - 822127136@ulife.com.br

823214064 – JÚLIA SILVA PEREIRA – ADS - 823214064@ulife.com.br

823126459 – RANGEL RIBEIRO SANTOS – ADS -823126459@ulife.com.br

8222241099 – VÍTOR DE SOUZA – CCP - 8222240199@ulife.com.br

SUMÁRIO

1 – TÉCNICAS DE TESTES ÁGEIS - TDD

1.1 – **1º Teste TDD**

Tamanho do vetor esperado menor que o vetor proposto.

```java
package exemplotdd;

public class ExemploTDD {



    /**
     * @param args
     * @method: main()
     */
    public static void main(String[] args) {
        OrdenaTest tdd = new OrdenaTest();
    }

}
```

```java
package exemplotdd;

class OrdenaTest {

    //constantes de cores para o console
    public static final String ANSI_RED_BACKGROUND = "\u001B[41m";
    public static final String ANSI_RESET = "\u001B[0m";
    public static final String ANSI_GREEN_BACKGROUND = "\u001B[42m";


    /**
     * @constructor: OrdenaTest()
     */
    public OrdenaTest(){
        boolean varBol1, varBol2;
        int proposto[] = new int[]{10,9};
        int esperado[] = new int[]{9,10};
        int inesperado[] = new int[]{9};

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

        varBol1 = caso1Test(proposto.length, inesperado.length);
        varBol2 = caso2Test(proposto, esperado);


```

```
27          System.out.println("""
28                                  Teste de Ordenação
29                                  ==================================""");
30
31          if(varBol1 == true && varBol2 == true){
32              System.out.println(ANSI_GREEN_BACKGROUND + "
33                      + "        " + ANSI_RESET);
34
35          } else {
36              System.out.println(ANSI_RED_BACKGROUND + "                                "
37                      + "        " + ANSI_RESET);
38          }
39          System.out.println("Ficou com o mesmo tamanho: "
40                      + caso1Test(proposto.length, inesperado.length));
41              System.out.println("Ordenou com sucesso: "
42                      + caso2Test(proposto, esperado));
43
44      }
45
46
47      /**
48       * @método: caso1Test()
49       */
50      private boolean caso1Test(int tamprop, int tamesp) {
51          boolean resp = true;
52          if( tamprop != tamesp) resp = false;
53          return resp;
54      }
```

```
55
56
57      /**
58       * @método: caso2Test()
59       */
60      private boolean caso2Test(int[] prop, int[] esp) {
61          return numerosIguais(prop, esp);
62      }
63
64
65      /**
66       * @método: numerosIguais()
67       */
68      private boolean numerosIguais(int nums1[], int nums2[]) {
69          boolean resultado = true;
70          for(int i=0,j=0;i < nums1.length; i++,j++){
71              if( nums1[i] != nums2[i]){
72                  resultado = false;
73                  i = nums1.length;
74              }
75          }
76          return resultado;
77      }
78
79  }
```

```
1   package exemplotdd;
2
3   class Ordena {
4
5       /**
6        * @constructor: Ordena()
7        */
8       public Ordena(){
9
10      }
11
12      void OrdenaNumerosCrescentes(int[] vetor) {
13      }
14
15  }
```

Exibição no console

```
run:
Teste de Ordenação
================================
[redacted]
Ficou com o mesmo tamanho: false
Ordenou com sucesso: false
BUILD SUCCESSFUL (total time: 0 seconds)
```

1.2 – 2º Teste TDD

Tamanho do vetor esperado igual ao proposto, mas não foi ordenado.

```java
package exemplotdd;

public class ExemploTDD {


    /**
     * @param args
     * @method: main()
     */
    public static void main(String[] args) {
        OrdenaTest tdd = new OrdenaTest();
    }

}
```

```java
package exemplotdd;

class OrdenaTest {

    //constantes de cores para o console
    public static final String ANSI_RED_BACKGROUND = "\u001B[41m";
    public static final String ANSI_RESET = "\u001B[0m";
    public static final String ANSI_GREEN_BACKGROUND = "\u001B[42m";


    /**
     * @constructor: OrdenaTest()
     */
    public OrdenaTest(){
        boolean varBol1, varBol2;
        int proposto[] = new int[]{10,9};
        int esperado[] = new int[]{9,10};
        int inesperado[] = new int[]{9};

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

        varBol1 = caso1Test(proposto.length, esperado.length);
        varBol2 = caso2Test(proposto, esperado);

```

```
26
27          System.out.println("""
28                                  Teste de Ordenação
29                                  ================================="""");
30
31          if(varBol1 == true && varBol2 == true){
32              System.out.println(ANSI_GREEN_BACKGROUND + "                          "
33                      + "          " + ANSI_RESET);
34
35          } else {
36              System.out.println(ANSI_RED_BACKGROUND + "                          "
37                      + "        " + ANSI_RESET);
38          }
39          System.out.println("Ficou com o mesmo tamanho: "
40                      + caso1Test(proposto.length, esperado.length));
41              System.out.println("Ordenou com sucesso: "
42                      + caso2Test(proposto, esperado));
43
44      }
45
46
```

```
47      /**
48       * @método: caso1Test()
49       */
50      private boolean caso1Test(int tamprop, int tamesp) {
51          boolean resp = true;
52          if( tamprop != tamesp) resp = false;
53          return resp;
54      }
55
56
57      /**
58       * @método: caso2Test()
59       */
60      private boolean caso2Test(int[] prop, int[] esp) {
61          return numerosIguais(prop, esp);
62      }
63
64
65      /**
66       * @método: numerosIguais()
67       */
68      private boolean numerosIguais(int nums1[], int nums2[]) {
69          boolean resultado = true;
70          for(int i=0,j=0;i < nums1.length; i++,j++){
71              if( nums1[i] != nums2[i]){
72                  resultado = false;
73                  i = nums1.length;
74              }
```

```
            }
            return resultado;
        }

    }
```

```
package exemplotdd;

class Ordena {

    /**
     * @constructor: Ordena()
     */
    public Ordena(){

    }

    void OrdenaNumerosCrescentes(int[] vetor) {
    }

}
```

Exibição no console

```
run:
Teste de Ordenação
==============================
[redacted]
Ficou com o mesmo tamanho: true
Ordenou com sucesso: false
BUILD SUCCESSFUL (total time: 0 seconds)
```

1.3 – 3º TESTE TDD – VERDADEIRO (FAKE)

Tamanho do vetor esperado igual ao proposto e ordenado.

```java
package exemplotdd;

public class ExemploTDD {


    /**
     * @param args
     * @method: main()
     */
    public static void main(String[] args) {
        OrdenaTest tdd = new OrdenaTest();
    }

}
```

```java
package exemplotdd;

class OrdenaTest {

    //constantes de cores para o console
    public static final String ANSI_RED_BACKGROUND = "\u001B[41m";
    public static final String ANSI_RESET = "\u001B[0m";
    public static final String ANSI_GREEN_BACKGROUND = "\u001B[42m";


    /**
     * @constructor: OrdenaTest()
     */
    public OrdenaTest(){
        boolean varBol1, varBol2;
        int proposto[] = new int[]{10,9};
        int esperado[] = new int[]{9,10};
        int inesperado[] = new int[]{9};

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

        varBol1 = caso1Test(proposto.length, esperado.length);
        varBol2 = caso2Test(proposto, esperado);

```

```
26
27          System.out.println("""
28                                  Teste de Ordenação
29                                  ================================="""");
30
31          if(varBol1 == true && varBol2 == true){
32              System.out.println(ANSI_GREEN_BACKGROUND + "                                "
33                      + "          " + ANSI_RESET);
34
35          } else {
36              System.out.println(ANSI_RED_BACKGROUND + "                                "
37                      + "        " + ANSI_RESET);
38          }
39          System.out.println("Ficou com o mesmo tamanho: "
40                      + caso1Test(proposto.length, esperado.length));
41              System.out.println("Ordenou com sucesso: "
42                      + caso2Test(proposto, esperado));
43
44      }
45
46
```

```
47      /**
48       * @método: caso1Test()
49       */
50      private boolean caso1Test(int tamprop, int tamesp) {
51          boolean resp = true;
52          if( tamprop != tamesp) resp = false;
53          return resp;
54      }
55
56
57      /**
58       * @método: caso2Test()
59       */
60      private boolean caso2Test(int[] prop, int[] esp) {
61          return numerosIguais(prop, esp);
62      }
63
64
65      /**
66       * @método: numerosIguais()
67       */
68      private boolean numerosIguais(int nums1[], int nums2[]) {
69          boolean resultado = true;
70          for(int i=0,j=0;i < nums1.length; i++,j++){
71              if( nums1[i] != nums2[i]){
72                  resultado = false;
73                  i = nums1.length;
74              }
```

```
            }
            return resultado;
        }

    }
```

```
package exemplotdd;

class Ordena {

    /**
     * @constructor: Ordena()
     */
    public Ordena(){

    }

    void OrdenaNumerosCrescentes(int[] vetor) {
        int rascunho;
        if(vetor[0] > vetor[1]){
            rascunho = vetor[1];
            vetor[1] = vetor[0];
            vetor[0] = rascunho;
        }
    }
}
```

Exibição no console

```
run:
Teste de Ordenação
================================
[green bar]
Ficou com o mesmo tamanho: true
Ordenou com sucesso: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

1.4 – 4º TESTE TDD – REFATORADO - VERDADEIRO

Tamanho do vetor esperado igual ao proposto e ordenado.

```java
package exemplotdd;

public class ExemploTDD {

    //private int[] vetorPreenchido;

    /**
     * @constructor: ExemploTDD()
     */
    public ExemploTDD(){
        int[] vetorPreenchido;

        PreencheVetor pvtdd = new PreencheVetor();
        vetorPreenchido = pvtdd.getArrayInt();
        OrdenaTest tdd = new OrdenaTest(vetorPreenchido);

    }


    /**
     * @param args
     * @method: main()
     */
    public static void main(String[] args) {
        ExemploTDD etdd = new ExemploTDD();
    }

}
```

```java
package exemplotdd;

import java.util.Arrays;
import javax.swing.JOptionPane;

class PreencheVetor {

    private int[] arrayInt; // array de inteiros

    /**
     * @constructor: PreencheVetor()
     */
    public PreencheVetor(){
        setArrayInt();
    }


```

```java
    /**
     * @method: getArrayInt()
     * @return the arrayInt
     * @objective: retornar o vetor
     */
    public int[] getArrayInt() {
        return arrayInt;
    }


    /**
     * @method: setArrayInt()
     * @param arrayInt the arrayInt to set
     * @objective: preencher o vetor com números inteiros
     */
    public final void setArrayInt() {

        int i;
        int tamArray;
        String saidaWhile = "entrada";

        while( "entrada".equals(saidaWhile) ){

            try{
                String tamVetor = JOptionPane.showInputDialog(null,"Informe o "
                        + "tamanho do vetor digitando um número inteiro: ");
                if(tamVetor != null){ // se não pressionou o botão cancelar....
                    tamArray = Integer.parseInt(tamVetor);
```

```
                    arrayInt = new int[tamArray];
                    System.out.println("Tamanho do vetor: " + tamArray);

                    for( i = 0; i < tamArray; i++){
                        try{
                            String entradaInt = JOptionPane.showInputDialog(""
                                    + "Digite número inteiro: ");
                            if(entradaInt != null){ // se não pressionou o botão
                                //cancelar....
                                arrayInt[i] = Integer.parseInt(entradaInt);
                            } else { System.exit(0);} // se pressionou o botão
                               //cancelar, o sistema é encerrado.

                        } catch(NumberFormatException ex){
                            JOptionPane.showMessageDialog(null, "Número inválido!"
                                    + "");
                            i--; // reter o índice na mesma posição;
                        }
                    }

                    if (i >= arrayInt.length){
                        System.out.println("Vetor: " + Arrays.toString(arrayInt));
                        saidaWhile = "saida"; // quebra o laço
                    }

                } else { System.exit(0);} // se pressionou o botão cancelar, o
                                //sistema é encerrado.

```

```
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(null, "Tamanho do vetor inválido!");
            }
        }
    }
}
```

```java
package exemplotdd;

class OrdenaTest {

    //constantes de cores para o console
    public static final String ANSI_RED_BACKGROUND = "\u001B[41m";
    public static final String ANSI_RESET = "\u001B[0m";
    public static final String ANSI_GREEN_BACKGROUND = "\u001B[42m";


    /**
     * @constructor: OrdenaTest()
     */
    public OrdenaTest(int[] vetorPreenchido){

        boolean varBol1, varBol2;
        int proposto[] = vetorPreenchido;
        int esperado[] = new int[]{2, 4, 9, 9, 14, 26, 28, 52, 63 , 120, 213};
        int inesperado[] = new int[]{9, 14, 26};

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

        // variáveis boleanas para condicionar as cores das faixas no console
        varBol1 = caso1Test(proposto.length, esperado.length);
        varBol2 = caso2Test(proposto, esperado);

```

```
28          System.out.println("""
29                                  Teste de Ordenação
30                                  ================================""");
31
32          if(varBol1 == true && varBol2 == true){
33              System.out.println(ANSI_GREEN_BACKGROUND + "                                  "
34                      + "          " + ANSI_RESET); // faixa vermelha
35
36          } else {
37              System.out.println(ANSI_RED_BACKGROUND + "                                  "
38                      + "        " + ANSI_RESET); // faixa verde
39          }
40
41          // exibindo as informações
42          System.out.println("Ficou com o mesmo tamanho: "
43                      + caso1Test(proposto.length, esperado.length));
44          System.out.println("Ordenou com sucesso: "
45                      + caso2Test(proposto, esperado));
46      }
47
48
49      /**
50       * @método: caso1Test()
51       */
52      private boolean caso1Test(int tamprop, int tamesp) {
53          boolean resp = true;
54          if( tamprop != tamesp) resp = false;
55          return resp;
56      }
```

```
28          System.out.println("""
29                                  Teste de Ordenação
30                                  ================================""");
31
32          if(varBol1 == true && varBol2 == true){
33              System.out.println(ANSI_GREEN_BACKGROUND + "                                  "
34                      + "          " + ANSI_RESET); // faixa vermelha
35
36          } else {
37              System.out.println(ANSI_RED_BACKGROUND + "                                  "
38                      + "        " + ANSI_RESET); // faixa verde
39          }
40
41          // exibindo as informações
42          System.out.println("Ficou com o mesmo tamanho: "
43                      + caso1Test(proposto.length, esperado.length));
44          System.out.println("Ordenou com sucesso: "
45                      + caso2Test(proposto, esperado));
46      }
47
48
```

```java
    /**
     * @método: caso1Test()
     */
    private boolean caso1Test(int tamprop, int tamesp) {
        boolean resp = true;
        if( tamprop != tamesp) resp = false;
        return resp;
    }


    /**
     * @método: caso2Test()
     */
    private boolean caso2Test(int[] prop, int[] esp) {
        return numerosIguais(prop, esp);
    }


    /**
     * @método: numerosIguais()
     */
    private boolean numerosIguais(int nums1[], int nums2[]) {
        boolean resultado = true;
        for(int i=0,j=0;i < nums1.length; i++,j++){
            if( nums1[i] != nums2[i]){
                resultado = false;
                i = nums1.length;
            }
        }
        return resultado;
    }

}
```

```java
package exemplotdd;

import java.util.Arrays;

class Ordena {


    /**
     * @constructor: Ordena()
     */
    public Ordena(){

    }


```

```java
    /**
     * @método: OrdenaNumerosCrescentes()
     * @objetivo: ordenar o vetor
     */
    void OrdenaNumerosCrescentes(int[] vetor) {

        int tamVetor = vetor.length; // tamanho do vetor
        int varTemp; // variável temporária

        for(int i=0; i < tamVetor; i++ ){
            for(int j=0; j < (tamVetor - i - 1); j++){
                if(vetor[j] > vetor[j+1]){
                    varTemp = vetor[j];
                    vetor[j] = vetor[j+1];
                    vetor[j+1] = varTemp;
                }
            }
        }
        System.out.println("Vetor ordenado: " + Arrays.toString(vetor) + "\n");
    }
}
```

```
run:
Tamanho do vetor: 11
Vetor: [26, 4, 9, 120, 63, 9, 52, 14, 2, 213, 28]
Vetor ordenado: [2, 4, 9, 9, 14, 26, 28, 52, 63, 120, 213]

Teste de Ordenação
==============================
[redacted]
Ficou com o mesmo tamanho: true
Ordenou com sucesso: true
BUILD SUCCESSFUL (total time: 36 seconds)
```

2 – TÉCNICA DE BDD

2.1 - Sistema de Validação de Usuário.

- Login
- Senha
- Botão Entrar

Acessa o SGBD – autorizada - > acessa.

- Inválida - > recusa.

2.3 – Perguntas dos DEVs

1 – Se o campo Login estiver vazio?

2 – O campo Login vai ser números, e-mail ou nome de usuário?

3 – Se o campo Login for números, quantos caracteres numéricos?

4 – Se o campo Login for nome de usuário, quantos caracteres alfabéticos deverá ter?

5 – Se o campo Login for numérico e entrar com letras?

6 – Se o campo Login for numérico e entrar com caracteres alfanuméricos?

7 – Se o campo Login for nome de usuário e entrar com números?

8 – Se o campo Login for nome de usuário e entrar com caracteres alfa numéricos?

9 – A senha vai ser numérica ou mista?

10 – De quantos caracteres será a composição da senha?

11 – Se o campo Senha for numérico e digitar caracteres alfa numéricos ou letras?

12 – Se o campo Senha estiver vazio?

13 – Se eu pressionar o botão Entrar e os campos Login ou Senha estiverem vazios ou se ambos estiverem vazios?

14 – Se os campos Login e Senha estiverem preenchidos e eu preencher o botão Entrar?

15 – Se o Login e Senha não estiverem cadastrados no SGBD?

16 – Se o Login e Senha estiverem cadastrados no SGBD?

2.3 – Respostas do PO

1 – O sistema não deverá aceitar campo vazio.

2 – O campo Login vai ser nome de usuário.

3 – O campo Login vai ser nome de usuário.

4 – O campo deverá permitir 20 caracteres.

5 – O campo login vai ser nome de usuário.

6 – O campo login vai ser nome de usuário.

7 – O campo Login deverá aceitar apenas espaços em branco e letras.

8 – O campo Login deverá aceitar apenas espaços em branco e letras.

9 – A senha vai ser mista, com letras maiúsculas, minúsculas, caracteres alfanuméricos e números.

10 – A senha será composta de 8 caracteres.

11 – A senha vai ser mista, com letras maiúsculas, minúsculas, caracteres alfanuméricos e números.

12 – O sistema não deverá aceitar campo Senha vazio.

13 – O botão Entrar deverá estar inativo quando os campos Login ou Senha e ambos estiverem vazios.

14 – O sistema irá verificar se usuário está cadastrado.

15 – O sistema nega a entrada e informa o usuário.

16 – O sistema autoriza a entrada do usuário.