Relatório da TAREFA #1 do Curso TDD (Coursera/ITA)

Aridio Gomes da Silva – aridiosilva@aridiosilva.com 11-OUT-2020

I - INTRODUÇÃO:

Este documento é resultado da elaboração da primeira tarefa prática valendo nota do Curso de Test-Driven Development (TDD) e que tem por objetivo descrever detalhadamente as etapas iterativas do processo de desenvolvimento do algoritmo, utilizando o TDD, para transformar uma cadeia de caracteres em formato Camel Case e uma lista de Strings com as palavras integrantes. A estrutura deste relatório segue o padrão solicitado na especificação da tarefa 1 a realizar.

II – CICLOS do DESENVOLVIMENTO (Utilizando TDD) do ALGORÍTMO

CICLO 1:

- Teste Adicionado:

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

import org.junit.Test;

public class TesteCamelCase {

	@Test
	public void testaListaVazia() {
		CamelCase cc = new CamelCase();
		assertTrue(cc.estaVazia());
		assertEquals(0, cc.tamanho());
	}
}
```

- Como Estava o Código Antes:

```
Public class CamelCase {

}
```

- Como Ficou o Código Depois:

```
public class CamelCase {

	public boolean estaVazia() {
		return true;
	}
	public int tamanho() {
		return 0;
	}
}
```

- Descrição do Que Foi Feito:

Na primeira iteração não existia a classe e comecei elaborando os dois primeiros testes, onde estarei verificando se a coleção List1<string> de palavras convertidas do CamelCase está vazia e o seu tamanho da lista deve ser zero. Desta forma rodando o caso de teste, que falhou porque os dois métodos não existiam ainda. Assim, alterei o código escrito para passar o teste, nesta fase preliminar de modo a poder construir em pequenos passos o algoritmo, pois no ciclo do TDD é primeiro escrever os testes e depois implementar o código para que o teste passe.

CICLO 2:

- Teste Adicionado:

```
	@Test
	public void testaCamelCaseUm() {
		CamelCase cc = new CamelCase();
		assertTrue(cc.IncluiTextoCamelCase("nome"));
		assertEquals("nome", cc.LePrimeiroItemLista());
	}
```

Relatório da TAREFA #1 do Curso TDD (Coursera/ITA)

Aridio Gomes da Silva – aridiosilva@aridiosilva.com 11-OUT-2020

- Como Estava o Código Antes:

```
public class CamelCase {
	public boolean estaVazia() {
		return true;
	}
	public int tamanho() {
		return 0;
	}
	public boolean IncluiTextoCamelCase(String string) {
		return false;
	}
	public Object LePrimeiroItemLista() {
		return null;
	}
}
```

- Como Ficou o Código Depois:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

	private List<String> list1 = new ArrayList<>(20);
	private int numItensLista = 0;

	public boolean estaVazia() {
		return numItensLista == 0;
	}
	public int tamanho() {
		return numItensLista;
	}
	public boolean IncluiTextoCamelCase(String texto) {
		list1.add(numItensLista, texto);
		numItensLista++;
		return true;
	}
	public String LePrimeiroItemLista() {
		return list1.get(0);
	}
}
```

- Descrição:

Na segunda iteração foi inserido o teste case (“nome”) cujo resultado deve ser (“nome”), e que envolve a chamada de dois métodos da Classe de Conversão do CamelCase que são o IncluiTextCamelCase (“nome”) e o método que LePrimeiroElemento da Lista e de forma que façam os testes falharem. E para permitir o teste foi criada coleção List1<string> e a variável numItensLista que indica o número de palavras encontradas no texto CamelCase. Assim, foi feita a modificação dos métodos de tamanho() da lista e estaVazia(), de modo a implementar código real já compatível com a evolução no momento do projeto do algoritmo. Seguindo assim a lógica do TDD de começar simples na lógica e ir incrementalmente ampliando, também, de forma simples, o algoritmo deste desafio do Cursera/ITA TDD.

CICLO 3:

- Teste Adicionado e Testes Alterados:

```
	private CamelCase cc;
	@Before
	public void inicializaCamelCase() {
	    cc = new CamelCase(20);
	}
	@Test
```

```
	public void testaCamelCaseUm() {
		// Teste 1 - String com uma palavra em CamelCase

		assertTrue(cc.IncluiTextoCamelCase("nome"));
		assertEquals("nome", cc.LeItemLista(1));
	}

	@Test
	public void testaCamelCaseDois() {
		// Teste 2 - String com Duas palavras em CamelCase

		assertTrue(cc.IncluiTextoCamelCase("nome"));
		assertEquals("nome", cc.LeItemLista(1));
		assertTrue(cc.IncluiTextoCamelCase("Composto"));
		assertEquals("composto", cc.LeItemLista(2));
	}}
```

- Como Estava o Código Antes:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

	private List<String> list1 = new ArrayList<>(20);
	private int numItensLista = 0;

	public boolean estaVazia() {
		return numItensLista == 0;
	}
	public int tamanho() {
		return numItensLista;
	}
	public boolean IncluiTextoCamelCase(String texto) {
		list1.add(numItensLista, texto);
		return true;
	}
	public String LePrimeiroItemLista() {
		return list1.get(0);
	}
}
```

- Como Ficou o Código Completo Depois:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

	private int maxItensLista;
	private List<String> list1 = new ArrayList<>(maxItensLista);
	private int numItensLista;

	public CamelCase(int maximo) {
		maxItensLista = maximo;
		numItensLista = 0;
	}
	public boolean estaVazia() {
		return numItensLista == 0;
	}
	public int tamanho() {
		return numItensLista;
	}
	public boolean IncluiTextoCamelCase(String texto) {
		if (numItensLista == maxItensLista) {
```

```
                    return false;   // lista cheia
             }
             list1.add(numItensLista, texto);
             numItensLista++;
             return true;
      }
      public String LeItemLista(int i) {
             if (i > maxItensLista + 1 || i < 1 ) {
                    // Gerar Error  IndexOutOfBoundsException
                    throw new IndexOutOfBoundsException("Indice fora da lista");
             }
             String s = list1.get(i - 1);
             return s.toLowerCase();
      }
}
```

- Descrição:

Na terceira iteração do TDD fiz a eliminação dos código duplicado de instanciação do objeto da classe CameCase, por meio da anotação @Before, e com isso, foram removidos dos métodos de teste o código redundante, e além disso foi criado um Construtor na Classe CamelCase de modo a inicializar a coleção Lista com um tamanho, inicializada a variável de tamanho máximo da lista por meio de parâmetro passado no construtor, e foi inicializado o número de itens da lista. Neste terceiro passo, foi criado o teste case que irá fornecer o texto em formato CamelCase com duas palavras, atendendo assim o texto case pedido nesta tarefa 1, sendo criado o método testaCamelCaseDois(); Foi eliminado o Método LePrimeiroItemLista() e substituído pelo método LeItemLista(). Foram feitos ajustes no códigos dos métodos de modo a tratar a variável de maxItensLista, e incluído os tratamentos e geração IndexOutOfBoundsException, quando o índice da palavra na coleção lista estiver fora dos limites inferior e superior. Além disso, foi feita a conversão das palavras lidas da Lista de uppercase para lowercase e assim atender a especificação desta tarefa 1 do curso CURSERA/ITA TDD. E o código final é mostrado acima de como ficou o código do algoritmo implementado até o presente momento.

CICLO 4:

- Teste Adicionado:

```
      public void testaCameCaseTres() {
             // Teste 3 - String "Nome"

             assertTrue(cc.IncluiTextoCamelCase("Nome"));
             assertEquals("nome", cc.LeItemLista(1));
      }
      public void testaCameCaseQuatro() {
             // Teste 4 - String "NomeCompleto"

             assertTrue(cc.IncluiTextoCamelCase("Nome"));
             assertEquals("nome", cc.LeItemLista(1));
             assertTrue(cc.IncluiTextoCamelCase("Composto"));
             assertEquals("composto", cc.LeItemLista(2));
      }
```

- Como Estava o Código Antes:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

      private int maxItensLista;
```

```
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;

        public CamelCase(int maximo) {
                maxItensLista = maximo;
                numItensLista = 0;
        }
        public boolean estaVazia() {
                return numItensLista == 0;
        }
        public int tamanho() {
                return numItensLista;
        }
        public boolean IncluiTextoCamelCase(String texto) {
                if (numItensLista == maxItensLista) {
                        return false;   // lista cheia
                }
                list1.add(numItensLista, texto);
                numItensLista++;
                return true;
        }
        public String LeItemLista(int i) {
                if (i > maxItensLista + 1 || i < 1 ) {
                // Gerar Error  IndexOutOfBoundsException
                  throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
                }
                String s = list1.get(i - 1);
                return s.toLowerCase();
        }
}
```

- Como Ficou o Código Depois:

```
Sem alteração no Código de implementação da solução, porque os dois casos de testes utilizam a estrutura do código escrito sem necessidade de alteração.
```

- Descrição:

Na quarta iteração do TDD oi feita a introdução de dois casos de testes no conjunto de casos de testes para poder testar as condições: (“Nome” cujo resultado deve ser “nome”) e (“NomeComposto” cujo resulado deve ser (“nome”,”composto”). O teste passou sem necessidade de alteração e ajustes no código do algoritmo sendo implementado.

CICLO 5:

- Teste Adicionado:

```
        public void testaCameCaseCinco() {
           // Teste 5 - String "CPF"

                assertTrue(cc.IncluiTextoCamelCase("CPF"));
                assertEquals("CPF", cc.LeItemLista(1));
        }
```

- Como Estava o Código Antes:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

        private int maxItensLista;
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;

        public CamelCase(int maximo) {
```

```
                maxItensLista = maximo;
                numItensLista = 0;
        }
        public boolean estaVazia() {
                return numItensLista == 0;
        }
        public int tamanho() {
                return numItensLista;
        }
        public boolean IncluiTextoCamelCase(String texto) {
                if (numItensLista == maxItensLista) {
                        return false;   // lista cheia
                }
                list1.add(numItensLista, texto);
                numItensLista++;
                return true;
        }
        public String LeItemLista(int i) {
                if (i > maxItensLista + 1 || i < 1 ) {
                // Gerar Error  IndexOutOfBoundsException
                  throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
                }
                String s = list1.get(i - 1);
                return s.toLowerCase();
        }
}
```

- Como Ficou o Código Depois:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

        private int maxItensLista;
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;

        public CamelCase(int maximo) {
                maxItensLista = maximo;
                numItensLista = 0;
        }
        public boolean estaVazia() {
                return numItensLista == 0;
        }
        public int tamanho() {
                return numItensLista;
        }
        public boolean stringCheck(String test1, String test2) {
            if ((test1 == null) || (test2 == null)) {
                 return false;
            }
            return test1.compareTo(test2) > 0;
        }

        public boolean ComparaStrings ( String s1, String s2) {
             return stringCheck(s1, s2);
        }
        public boolean IncluiTextoCamelCase(String texto) {
                if (numItensLista == maxItensLista) {
                        return false;   // lista cheia
                }
                if (ComparaStrings (texto, texto.toUpperCase())) {
                        // word toda em uppercase
                        list1.add(numItensLista, texto);
                } else {
```

```
                        list1.add(numItensLista, texto.toLowerCase() );
                }
                numItensLista++;
                return true;
        }
        public String LeItemLista(int i) {
                if (i > maxItensLista + 1 || i < 1 ) {
                        // Gerar Error  IndexOutOfBoundsException
                        throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
                }
                return list1.get(i - 1);
        }
}
```

- Descrição:

Na quinta iteração do TDD foi criado o caso de teste (“CPF”) e o resultado deve ser (“CPF”), o que fez exigiu o tratamento da condição em que palavras todas em maiúsculas devem ser preservadas como tal para o processo de conversão final. Assim foram criados dois métodos stringCheck() e ComparaStrings de modo a checar se a o string está em maiúscula e assim retornar um indicador para o Metodo IncluiTextoCamelCase() e assim ele inserir a palavra na coleção lista em maiúscula, e assim foi ajustes no Método LeItemLista() de modo a não converter todas as palavras lidas da coleção lista para minúscula, porque a Lista<string> já estará agora com as palavras no final final requerido pelas especificações desta tarefa 1 do curso CURSERA/ITA TDD e o código final ficou como indicado acima.

CICLO 6:

- Teste Adicionado:

```
        public void testaCameCaseSeis() {
                // Teste 6 - String "numeroCPF"

                assertTrue(cc.IncluiTextoCamelCase("numero"));
                assertEquals("numero", cc.LeItemLista(1));
                assertTrue(cc.IncluiTextoCamelCase("CPF"));
                assertEquals("CPF", cc.LeItemLista(2));
        }
```

- Como Estava o Código Antes:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

        private int maxItensLista;
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;

        public CamelCase(int maximo) {
                maxItensLista = maximo;
                numItensLista = 0;
        }

        public boolean estaVazia() {
                return numItensLista == 0;
        }

        public int tamanho() {
                return numItensLista;
        }

        public boolean stringCheck(String test1, String test2) {
            if ((test1 == null) || (test2 == null)) {
                return false;
```

```
            }
            return test1.compareTo(test2) > 0;
        }

        public boolean ComparaStrings ( String s1, String s2) {
             return stringCheck(s1, s2);
    }

        public boolean IncluiTextoCamelCase(String texto) {
                if (numItensLista == maxItensLista) {
                        return false;   // lista cheia
                }
                if (ComparaStrings (texto, texto.toUpperCase())) {
                        // word toda em uppercase
                        list1.add(numItensLista, texto);
                } else {
                                list1.add(numItensLista, texto.toLowerCase() );
                }
                numItensLista++;
                return true;
        }

        public String LeItemLista(int i) {
                if (i > maxItensLista + 1 || i < 1 ) {
                        // Gerar Error  IndexOutOfBoundsException
                        throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
                }
                return list1.get(i - 1);
        }
}
```

- Como Ficou o Código Depois:

```
Sem alteração no Código de implementação da solução, porque os dois casos de testes utilizam a estrutura do código escrito sem necessidade de alteração
```

- Descrição:

Na sexta iteração do TDD foi criado o sexto caso de teste (“numeroCPF”) e o resultrado deve ser (“numero”,CPF”), o teste passou sem necessidade de alteração e ajustes no código do algoritmo sendo implementado, pois este teste usar os métodos do código do algoritmo sendo desenvolvido sem nenhuma necessidade de alteração ou ajustes.

CICLO 7:

- Teste Adicionado:

```
        public void testaCameCaseSete() {
                // Teste 7 - String "numeroCPFContribuinte"

                assertTrue(cc.IncluiTextoCamelCase("numero"));
                assertEquals("numero", cc.LeItemLista(1));
                assertTrue(cc.IncluiTextoCamelCase("CPF"));
                assertEquals("CPF", cc.LeItemLista(2));
                assertTrue(cc.IncluiTextoCamelCase("Contribuinte"));
                assertEquals("contribuinte", cc.LeItemLista(3));
        }
```

- Como Estava o Código Antes:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {
```

```
        private int maxItensLista;
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;

        public CamelCase(int maximo) {
                maxItensLista = maximo;
                numItensLista = 0;
        }

        public boolean estaVazia() {
                return numItensLista == 0;
        }

        public int tamanho() {
                return numItensLista;
        }

        public boolean stringCheck(String test1, String test2) {
            if ((test1 == null) || (test2 == null)) {
                return false;
            }
            return test1.compareTo(test2) > 0;
        }

        public boolean ComparaStrings ( String s1, String s2) {
             return stringCheck(s1, s2);
    }

        public boolean IncluiTextoCamelCase(String texto) {
                if (numItensLista == maxItensLista) {
                        return false;   // lista cheia
                }
                if (ComparaStrings (texto, texto.toUpperCase())) {
                        // word toda em uppercase
                        list1.add(numItensLista, texto);
                } else {
                                list1.add(numItensLista, texto.toLowerCase() );
                }
                numItensLista++;
                return true;
        }

        public String LeItemLista(int i) {
                if (i > maxItensLista + 1 || i < 1 ) {
                        // Gerar Error  IndexOutOfBoundsException
                        throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
                }
                return list1.get(i - 1);
        }
}
```

- Como Ficou o Código Depois:

```
Sem alteração no Código de implementação da solução, porque os dois casos de testes utilizam a estrutura do código escrito sem necessidade de alteração
```

- Descrição:

Na sétima iteração do TDD foi criado o sexto caso de teste (“numeroCPFContribuinte” e o resultado do teste deve ser “numero”,”CPF”,”contribuinte”) – conforme determinado na especificação desta Tarefa 1. o teste passou sem necessidade de alteração e ajustes no código do algoritmo sendo implementado, pois este teste usar os métodos do código do algoritmo sendo desenvolvido sem nenhuma necessidade de alteração ou ajustes.

CICLO 8:

- Teste Adicionado:

```
        public void testaCameCaseSete() {
                // Teste 7 - String "numeroCPFContribuinte"

                assertTrue(cc.IncluiTextoCamelCase("numero"));
                assertEquals("numero", cc.LeItemLista(1));
                assertTrue(cc.IncluiTextoCamelCase("CPF"));
                assertEquals("CPF", cc.LeItemLista(2));
                assertTrue(cc.IncluiTextoCamelCase("Contribuinte"));
                assertEquals("contribuinte", cc.LeItemLista(3));
        }
```

- Como Estava o Código Antes:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

        private int maxItensLista;
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;

        public CamelCase(int maximo) {
                maxItensLista = maximo;
                numItensLista = 0;
        }

        public boolean estaVazia() {
                return numItensLista == 0;
        }

        public int tamanho() {
                return numItensLista;
        }

        public boolean stringCheck(String test1, String test2) {
            if ((test1 == null) || (test2 == null)) {
                return false;
            }
            return test1.compareTo(test2) > 0;
        }

        public boolean ComparaStrings ( String s1, String s2) {
             return stringCheck(s1, s2);
    }

        public boolean IncluiTextoCamelCase(String texto) {
                if (numItensLista == maxItensLista) {
                        return false;   // lista cheia
                }
                if (ComparaStrings (texto, texto.toUpperCase())) {
                        // word toda em uppercase
                        list1.add(numItensLista, texto);
                } else {
                                list1.add(numItensLista, texto.toLowerCase() );
                }
                numItensLista++;
                return true;
        }

        public String LeItemLista(int i) {
                if (i > maxItensLista + 1 || i < 1 ) {
                        // Gerar Error  IndexOutOfBoundsException
                        throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
```

```
                }
                return list1.get(i - 1);
        }
}
```

- Como Ficou o Código Depois:

```
Sem alteração no Código de implementação da solução, porque os dois casos de testes utilizam a estrutura do código escrito sem necessidade de alteração
```

- Descrição:

Na oitava iteração do TDD foi criado o sexto caso de teste (“numeroCPFCOntribuinte” e o resultado do teste deve ser “numero”,”CPF”,”contribuinte”) – conforme determinado na especificação desta Tarefa 1. o teste passou sem necessidade de alteração e ajustes no código do algoritmo sendo implementado, pois este teste usar os métodos do código do algoritmo sendo desenvolvido sem nenhuma necessidade de alteração ou ajustes.

CICLO 9:

- Teste Adicionado:

```
        public void testaCameCaseOito() {
                // Teste 8 - String "recupera10Primeiros"

                assertTrue(cc.IncluiTextoCamelCase("recupera"));
                assertEquals("recupera", cc.LeItemLista(1));
                assertTrue(cc.IncluiTextoCamelCase("10"));
                assertEquals("10", cc.LeItemLista(2));
                assertTrue(cc.IncluiTextoCamelCase("Primeiros"));
                assertEquals("Primeiros", cc.LeItemLista(3));
        }
```

- Como Estava o Código Antes:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

        private int maxItensLista;
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;

        public CamelCase(int maximo) {
                maxItensLista = maximo;
                numItensLista = 0;
        }

        public boolean estaVazia() {
                return numItensLista == 0;
        }

        public int tamanho() {
                return numItensLista;
        }

        public boolean stringCheck(String test1, String test2) {
            if ((test1 == null) || (test2 == null)) {
                return false;
            }
            return test1.compareTo(test2) > 0;
        }

        public boolean ComparaStrings ( String s1, String s2) {
```

```
            return stringCheck(s1, s2);
    }

        public boolean IncluiTextoCamelCase(String texto) {
                if (numItensLista == maxItensLista) {
                        return false;   // lista cheia
                }
                if (ComparaStrings (texto, texto.toUpperCase())) {
                        // word toda em uppercase
                        list1.add(numItensLista, texto);
                } else {
                                list1.add(numItensLista, texto.toLowerCase() );
                }
                numItensLista++;
                return true;
        }

        public String LeItemLista(int i) {
                if (i > maxItensLista + 1 || i < 1 ) {
                        // Gerar Error  IndexOutOfBoundsException
                        throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
                }
                return list1.get(i - 1);
        }
}
```

- Como Ficou o Código Depois:

```
Sem alteração no Código de implementação da solução, porque os dois casos de testes utilizam a estrutura do código escrito sem necessidade de alteração
```

- Descrição:

Na oitava iteração do TDD foi criado o sexto caso de teste (“Recupera10Primeiros” e o resultado do teste deve ser “recupera”,”10”,”primeiros”) – conforme determinado na especificação desta Tarefa 1. o teste passou sem necessidade de alteração e ajustes no código do algoritmo sendo implementado, pois este teste usar os métodos do código do algoritmo sendo desenvolvido sem nenhuma necessidade de alteração ou ajustes.

CICLO 10:

- Teste Adicionado:

```
        public void testaCameCaseNove() {
            // Teste 9 - String "10Primeiros"
            assertFalse(cc.IncluiTextoCamelCase("10Primeiros"));
            assertTrue(cc.IncluiTextoCamelCase("Primeiros"));
            assertEquals("primeiros", cc.LeItemLista(1));
        }
```

- Como Estava o Código Antes:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

        private int maxItensLista;
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;

        public CamelCase(int maximo) {
                maxItensLista = maximo;
```

```
                numItensLista = 0;
        }

        public boolean estaVazia() {
                return numItensLista == 0;
        }

        public int tamanho() {
                return numItensLista;
        }

        public boolean stringCheck(String test1, String test2) {
            if ((test1 == null) || (test2 == null)) {
                return false;
            }
            return test1.compareTo(test2) > 0;
        }

        public boolean ComparaStrings ( String s1, String s2) {
             return stringCheck(s1, s2);
    }

        public boolean IncluiTextoCamelCase(String texto) {
                if (numItensLista == maxItensLista) {
                        return false;   // lista cheia
                }
                if (ComparaStrings (texto, texto.toUpperCase())) {
                        // word toda em uppercase
                        list1.add(numItensLista, texto);
                } else {
                                list1.add(numItensLista, texto.toLowerCase() );
                }
                numItensLista++;
                return true;
        }

        public String LeItemLista(int i) {
                if (i > maxItensLista + 1 || i < 1 ) {
                        // Gerar Error  IndexOutOfBoundsException
                        throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
                }
                return list1.get(i - 1);
        }
}
```

- Como Ficou o Código Depois:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

        private int maxItensLista;
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;

        public CamelCase(int maximo) {
                maxItensLista = maximo;
                numItensLista = 0;
        }
        public boolean estaVazia() {
                return numItensLista == 0;
        }
        public int tamanho() {
                return numItensLista;
        }
        public boolean stringCheck(String test1, String test2) {
```

```
            if ((test1 == null) || (test2 == null)) {
                return false;
            }
            return test1.compareTo(test2) > 0;
        }
        public boolean ComparaStrings ( String s1, String s2) {
             return stringCheck(s1, s2);
        }
        public boolean IncluiTextoCamelCase(String texto) {
                if (text.matches("[0-9]+") && text.length() > 2) {
                        return false;
                }
                if (numItensLista == maxItensLista) {
                        return false;   // lista cheia
                }
                if (ComparaStrings (texto, texto.toUpperCase())) {
                        list1.add(numItensLista, texto); // word toda em uppercase
                } else {
                             list1.add(numItensLista, texto.toLowerCase() );
                }
                numItensLista++;
                return true;
        }
        public String LeItemLista(int i) {
                if (i > maxItensLista + 1 || i < 1 ) {
                        // Gerar Error  IndexOutOfBoundsException
                        throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
                }
                return list1.get(i - 1);
        }
}
```

- Descrição:

Na nova iteração do TDD foi criado o sexto caso de teste (“10Primeiros” e o resultado deve ser Inválido Não Deve Começar com Números”, conforme determinado na especificação desta Tarefa 1. Para atender a esta restrição foi necessário alterar o código do Método IncluiTextoCamelCase() a fim de testar se o texto é constituído apenas de caracteres numéricos. Assim, o código do algoritmos resultante ficou como indicado acima.

CICLO 11:

- Teste Adicionado:

```
        public void testaCameCaseDez() {
                // Teste 10 - String nome#Composto"
                assertFalse(cc.IncluiTextoCamelCase("nome"));
                assertEquals("nome", cc.LeItemLista(1));
                assertFalse(cc.IncluiTextoCamelCase("#"));
                assertFalse(cc.IncluiTextoCamelCase("Composto"));
                assertEquals("composto", cc.LeItemLista(2));
        }
```

- Como Estava o Código Antes:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

        private int maxItensLista;
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;
```

```
        public CamelCase(int maximo) {
               maxItensLista = maximo;
               numItensLista = 0;
        }
        public boolean estaVazia() {
               return numItensLista == 0;
        }
        public int tamanho() {
               return numItensLista;
        }
        public boolean stringCheck(String test1, String test2) {
            if ((test1 == null) || (test2 == null)) {
                 return false;
            }
            return test1.compareTo(test2) > 0;
        }
        public boolean ComparaStrings ( String s1, String s2) {
             return stringCheck(s1, s2);
        }
        public boolean IncluiTextoCamelCase(String texto) {
               if (text.matches("[0-9]+") && text.length() > 2) {
                      return false;
               }
               if (numItensLista == maxItensLista) {
                      return false;   // lista cheia
               }
               if (ComparaStrings (texto, texto.toUpperCase())) {
                      list1.add(numItensLista, texto); // word toda em uppercase
               } else {
                            list1.add(numItensLista, texto.toLowerCase() );
               }
               numItensLista++;
               return true;
        }
        public String LeItemLista(int i) {
               if (i > maxItensLista + 1 || i < 1 ) {
                   // Gerar Error  IndexOutOfBoundsException
                   throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
               }
               return list1.get(i - 1);
        }
}
```

- Como Ficou o Código Depois:

```
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

        private int maxItensLista;
        private List<String> list1 = new ArrayList<>(maxItensLista);
        private int numItensLista;

        public CamelCase(int maximo) {
               maxItensLista = maximo;
               numItensLista = 0;
        }
        public boolean estaVazia() {
               return numItensLista == 0;
```

```
        }
        public int tamanho() {
               return numItensLista;
        }
        public boolean stringCheck(String test1, String test2) {
            if ((test1 == null) || (test2 == null)) {
                return false;
            }
            return test1.compareTo(test2) > 0;
        }
        public boolean ComparaStrings ( String s1, String s2) {
             return stringCheck(s1, s2);
    }
        public boolean veSeCaracteresEspeciais (String ss) {
            String n = ".*[0-9].*";
            String a = ".*[A-Z].*";
            return ss.matches(n) && ss.matches(a);
        }
        public boolean IncluiTextoCamelCase(String texto) {
               if (veSeCaracteresEspeciais (texto)) {
                      return false;   // Não pode conter caracteres especiais
               }
               if (texto.matches("[0-9]+") && texto.length() > 2) {
                      return false;   // Não pode conter só números
               }
               if (numItensLista == maxItensLista) {
                      return false;   // lista cheia
               }
               if (ComparaStrings (texto, texto.toUpperCase())) {
                      list1.add(numItensLista, texto); // word toda em uppercase
               } else {
                         list1.add(numItensLista, texto.toLowerCase() );
               }
               numItensLista++;
               return true;
        }
        public String LeItemLista(int i) {
               if (i > maxItensLista + 1 || i < 1 ) {
                 // Gerar Error  IndexOutOfBoundsException
                  throw new IndexOutOfBoundsException("Indice fora da lista e Inacessivel");
               }
               return list1.get(i - 1);
        }
}
```

- Descrição:

Na décima iteração do TDD foi criado o sexto caso de teste (“nome#Composto” e o resultado deve ser Inválido pois caracteres especiais não são permitidos”, conforme determinado na especificação desta Tarefa 1. Para atender a esta restrição foi necessário criado um novo método para analisar a presença de caracter especial no texto denominado veSeCaracteresEspeciais() e foi alterado o código do Método IncluiTextoCamelCase() a fim de chamar o novo método para ver se tem caracteres não permitidos no texto do CamelCase. Assim, o código do algoritmos resultante ficou como indicado acima.