CamelCase - Exercício TDD

1.1- Criação do teste do método de conversão.

```
package camelCase;

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

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

import org.junit.Before;
import org.junit.jupiter.api.Test;

class TestCamelCase {

	private CamelCase converter;

	@Before
	public void inicializaCamelCase() {
		converter = new CamelCase();
	}

	@Test
	void converterCamelCase() {
		List res = converter.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

}
```

1.2- Criação do método de conversão que falhará no teste.

```
package camelCase;

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

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		return new ArrayList();
	}
}
```

1.3- Falha no teste e código.

```
package camelCase;

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

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> lista = new ArrayList<String>();

		lista.add(original);

		return lista;
	}
}
```

1.4- Teste passando

2

2.1- Criação de teste com palavra que têm somente a primeira letra maiúscula, teste falhando.

```java
package camelCase;

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

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

import org.junit.jupiter.api.Test;

class TestCamelCase {

	@Test
	void converterCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	void converterSomentePrimeiraLetraMaiuscula() {
		List<String> res = CamelCase.converterCamelCase("Camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}
}
```

2.2 - Alterando código de maneira simples para que passe no teste.

```java
package camelCase;

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

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> lista = new ArrayList<String>();

		lista.add(original.toLowerCase());

		return lista;
	}
}
```

3.1-Adicionando teste com nome composto, não está passando.

```java
package camelCase;

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

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

import org.junit.jupiter.api.Test;

class TestCamelCase {

	@Test
	void converterCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	void converterSomentePrimeiraLetraMaiuscula() {
		List<String> res = CamelCase.converterCamelCase("Camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	void converterNomeComposto() {
		List<String> res =
CamelCase.converterCamelCase("camelCase");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");
		expected.add("Case");

		assertEquals(res, expected);
	}
}
```

3.2-Código para gerar lista que separe as palavras por letra maiúscula e a primeira esteja

em caixa baixa.

```
package camelCase;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		String words[] = original.split("(?=[A-Z])", -1);

		words[0] = words[0].toLowerCase();

		ArrayList<String> lista = new
ArrayList<String>(Arrays.asList(words));

		return lista;
	}
}
```

4.1-Refatorando classe, adicionando testes e métodos de separar por letras maiúsculas, e

setar primeira palavra em caixa baixa.

```
package camelCase;

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

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

import org.junit.jupiter.api.Test;

class TestCamelCase {

	@Test
	void converterCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	void converterSomentePrimeiraLetraMaiuscula() {
		List<String> res = CamelCase.converterCamelCase("Camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	void converterNomeComposto() {
		List<String> res =
CamelCase.converterCamelCase("camelCase");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");
		expected.add("Case");

		assertEquals(res, expected);
	}
```

```
	@Test
	void getWordsArray() {
		String res[] = CamelCase.getWordsArray("CamelCaseTeste");

		assertEquals(res[0], "camel");
		assertEquals(res[1], "Case");
		assertEquals(res[2], "Teste");
	}

	@Test
	void separarPorLetraMaiuscula() {
		String res[] =
CamelCase.separarPorLetraMaiuscula("camelCaseTeste");

		assertEquals(res[0], "camel");
		assertEquals(res[1], "Case");
		assertEquals(res[2], "Teste");
	}

	@Test
	void handleFirstWord() {
		String words[] = {"TesTe"};

		String res[] = CamelCase.handleFirstWord(words);

		assertEquals(res[0], "teste");
	}
}
```

4.2- Adicionando código para passar nos testes

```
package camelCase;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> lista = new
ArrayList<String>(Arrays.asList(getWordsArray(original)));

		return lista;
	}
```

```
	public static String[] handleFirstWord(String[] words) {
		words[0] = words[0].toLowerCase();

		return words;
	}

	public static String[] getWordsArray(String original) {
		String words[] = separarPorLetraMaiuscula(original);

		return handleFirstWord(words);
	}

	public static String[] separarPorLetraMaiuscula(String original) {

		String splited[] = original.split("(?=[A-Z])", -1);

		return splited;
	}
}
```

5.1- Adicionando teste de palavra em caixa alta, não está passando.

```
package camelCase;

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

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

import org.junit.jupiter.api.Test;

class TestCamelCase {

	@Test
	void converterCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
```

```
	void converterSomentePrimeiraLetraMaiuscula() {
		List<String> res = CamelCase.converterCamelCase("Camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	void converterNomeComposto() {
		List<String> res =
CamelCase.converterCamelCase("camelCase");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");
		expected.add("Case");

		assertEquals(res, expected);
	}

	@Test
	void getWordsArray() {
		String res[] = CamelCase.getWordsArray("CamelCaseTeste");

		assertEquals(res[0], "camel");
		assertEquals(res[1], "Case");
		assertEquals(res[2], "Teste");
	}

	@Test
	void separarPorLetraMaiuscula() {
		String res[] =
CamelCase.separarPorLetraMaiuscula("camelCaseTeste");

		assertEquals(res[0], "camel");
		assertEquals(res[1], "Case");
		assertEquals(res[2], "Teste");
	}

	@Test
	void handleFirstWord() {
		String words[] = {"TesTe"};

		String res[] = CamelCase.handleFirstWord(words);
```

```
		assertEquals(res[0], "teste");
	}

	@Test
	void converterPalavraCaixaAlta() {
		List<String> res = CamelCase.converterCamelCase("CPF");

		List<String> expected = new ArrayList<String>();
		expected.add("CPF");

		assertEquals(res, expected);
	}
}
```

5.2- Ajuste para que palavras como CPF sejam validadas.

```
package camelCase;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> lista = new
ArrayList<String>(Arrays.asList(getWordsArray(original)));

		return lista;
	}

	public static String[] handleFirstWord(String[] words) {
		if (words[0].toUpperCase() != words[0]) {
			words[0] = words[0].toLowerCase();
		}

		return words;
	}

	public static String[] getWordsArray(String original) {
		String words[] = separarPorLetraMaiuscula(original);

		return handleFirstWord(words);
```

```
	}

	public static String[] separarPorLetraMaiuscula(String original) {
		String splited[] = original.split("(?<=[a-z](?=[A-Z]))");

		return splited;
	}
}
```

6.1- Adicionando testes de palavras como nomeCPF, já passa no teste

```
package camelCase;

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

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

import org.junit.jupiter.api.Test;

class TestCamelCase {

	@Test
	void converterCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	void converterSomentePrimeiraLetraMaiuscula() {
		List<String> res = CamelCase.converterCamelCase("Camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	void converterNomeComposto() {
		List<String> res =
```

```
CamelCase.converterCamelCase("camelCase");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");
		expected.add("Case");

		assertEquals(res, expected);
	}

	@Test
	void getWordsArray() {
		String res[] = CamelCase.getWordsArray("CamelCaseTeste");

		assertEquals(res[0], "camel");
		assertEquals(res[1], "Case");
		assertEquals(res[2], "Teste");
	}

	@Test
	void separarPorLetraMaiuscula() {
		String res[] =
CamelCase.separarPorLetraMaiuscula("camelCaseTeste");

		assertEquals(res[0], "camel");
		assertEquals(res[1], "Case");
		assertEquals(res[2], "Teste");
	}

	@Test
	void handleFirstWord() {
		String words[] = {"TesTe"};

		String res[] = CamelCase.handleFirstWord(words);

		assertEquals(res[0], "teste");
	}

	@Test
	void converterPalavraCaixaAlta() {
		List<String> res = CamelCase.converterCamelCase("CPF");

		List<String> expected = new ArrayList<String>();
		expected.add("CPF");

		assertEquals(res, expected);
```

```
	}

	@Test
	void converterPalavraCaixaAltaCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("nomeCPF");

		List<String> expected = new ArrayList<String>();
		expected.add("nome");
		expected.add("CPF");

		assertEquals(res, expected);
	}
}
```

7.1 Adicionando teste para palavras como numeroCPFContribuinte, teste não passa.

```
package camelCase;

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

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

import org.junit.jupiter.api.Test;

class TestCamelCase {

	@Test
	void converterCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	void converterSomentePrimeiraLetraMaiuscula() {
		List<String> res = CamelCase.converterCamelCase("Camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");
```

```
            assertEquals(res, expected);
      }

      @Test
      void converterNomeComposto() {
            List<String> res =
CamelCase.converterCamelCase("camelCase");

            List<String> expected = new ArrayList<String>();
            expected.add("camel");
            expected.add("Case");

            assertEquals(res, expected);
      }

      @Test
      void getWordsArray() {
            String res[] = CamelCase.getWordsArray("CamelCaseTeste");

            assertEquals(res[0], "camel");
            assertEquals(res[1], "Case");
            assertEquals(res[2], "Teste");
      }

      @Test
      void separarPorLetraMaiuscula() {
            String res[] =
CamelCase.separarPorLetraMaiuscula("camelCaseTeste");

            assertEquals(res[0], "camel");
            assertEquals(res[1], "Case");
            assertEquals(res[2], "Teste");
      }

      @Test
      void handleFirstWord() {
            String words[] = {"TesTe"};

            String res[] = CamelCase.handleFirstWord(words);

            assertEquals(res[0], "teste");
      }

      @Test
```

```
	void converterPalavraCaixaAlta() {
		List<String> res = CamelCase.converterCamelCase("CPF");

		List<String> expected = new ArrayList<String>();
		expected.add("CPF");

		assertEquals(res, expected);
	}

	@Test
	void converterPalavraCaixaAltaCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("nomeCPF");

		List<String> expected = new ArrayList<String>();
		expected.add("nome");
		expected.add("CPF");

		assertEquals(res, expected);
	}

	@Test
	void converterPalavraNumeroCPFContribuinte() {
		List<String> res =
CamelCase.converterCamelCase("numeroCPFContribuinte");

		List<String> expected = new ArrayList<String>();
		expected.add("numero");
		expected.add("CPF");
		expected.add("Contribuinte");

		assertEquals(res, expected);
	}
}
```

7.2 - Alterando regex que separe as palavras.

```
package camelCase;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> lista = new
```

```
ArrayList<String>(Arrays.asList(getWordsArray(original)));

		return lista;
	}

	public static String[] handleFirstWord(String[] words) {
		if (words[0].toUpperCase() != words[0]) {
			words[0] = words[0].toLowerCase();
		}

		return words;
	}

	public static String[] getWordsArray(String original) {
		String words[] = separarPorLetraMaiuscula(original);

		return handleFirstWord(words);
	}

	public static String[] separarPorLetraMaiuscula(String original) {
		String splited[] =
original.split("(?<=[a-z](?=[A-Z]))|((?=[A-Z][a-z]))");

		return splited;
	}
}
```

8.1- Adicionando deste de palavras como recupera10Primeiros, teste não está passando

```
package camelCase;

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

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

import org.junit.jupiter.api.Test;

class TestCamelCase {

	@Test
	void converterCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");
```

```
            assertEquals(res, expected);
      }

      @Test
      void converterSomentePrimeiraLetraMaiuscula() {
            List<String> res = CamelCase.converterCamelCase("Camel");

            List<String> expected = new ArrayList<String>();
            expected.add("camel");

            assertEquals(res, expected);
      }

      @Test
      void converterNomeComposto() {
            List<String> res =
CamelCase.converterCamelCase("camelCase");

            List<String> expected = new ArrayList<String>();
            expected.add("camel");
            expected.add("Case");

            assertEquals(res, expected);
      }

      @Test
      void getWordsArray() {
            String res[] = CamelCase.getWordsArray("CamelCaseTeste");

            assertEquals(res[0], "camel");
            assertEquals(res[1], "Case");
            assertEquals(res[2], "Teste");
      }

      @Test
      void separarPorLetraMaiuscula() {
            String res[] =
CamelCase.separarPorLetraMaiuscula("camelCaseTeste");

            assertEquals(res[0], "camel");
            assertEquals(res[1], "Case");
            assertEquals(res[2], "Teste");
      }
```

```java
	@Test
	void handleFirstWord() {
		String words[] = {"TesTe"};

		String res[] = CamelCase.handleFirstWord(words);

		assertEquals(res[0], "teste");
	}

	@Test
	void converterPalavraCaixaAlta() {
		List<String> res = CamelCase.converterCamelCase("CPF");

		List<String> expected = new ArrayList<String>();
		expected.add("CPF");

		assertEquals(res, expected);
	}

	@Test
	void converterPalavraCaixaAltaCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("nomeCPF");

		List<String> expected = new ArrayList<String>();
		expected.add("nome");
		expected.add("CPF");

		assertEquals(res, expected);
	}

	@Test
	void converterPalavraNumeroCPFContribuinte() {
		List<String> res =
CamelCase.converterCamelCase("numeroCPFContribuinte");

		List<String> expected = new ArrayList<String>();
		expected.add("numero");
		expected.add("CPF");
		expected.add("Contribuinte");

		assertEquals(res, expected);
	}

	@Test
	void converterPalavraRecupera10Primeiros() {
```

```
            List<String> res =
CamelCase.converterCamelCase("recupera10Primeiros");

            List<String> expected = new ArrayList<String>();
            expected.add("recupera");
            expected.add("10");
            expected.add("Primeiros");

            assertEquals(res, expected);
      }
}
```

8.2-Alterando regex para que separador inclua números

```
package camelCase;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CamelCase {
      public static List<String> converterCamelCase(String original) {
            ArrayList<String> lista = new
ArrayList<String>(Arrays.asList(getWordsArray(original)));

            return lista;
      }

      public static String[] handleFirstWord(String[] words) {
            if (words[0].toUpperCase() != words[0]) {
                  words[0] = words[0].toLowerCase();
            }

            return words;
      }

      public static String[] getWordsArray(String original) {
            String words[] = separarPorLetraMaiuscula(original);

            return handleFirstWord(words);
      }

      public static String[] separarPorLetraMaiuscula(String original) {
            String splited[] =
```

```
original.split("(?<=[a-z](?=[A-Z0-10{1,}]))|((?=[A-Z][a-z]))");

            return splited;
      }
}
```

9.1-Testando string inválida, começando com número, teste não está passando.

```
package camelCase;

import org.junit.Test;

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

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

public class TestCamelCase {

      @Test
      public void converterCaixaBaixa() {
            List<String> res = CamelCase.converterCamelCase("camel");

            List<String> expected = new ArrayList<String>();
            expected.add("camel");

            assertEquals(res, expected);
      }

      @Test
      public void converterSomentePrimeiraLetraMaiuscula() {
            List<String> res = CamelCase.converterCamelCase("Camel");

            List<String> expected = new ArrayList<String>();
            expected.add("camel");

            assertEquals(res, expected);
      }

      @Test
      public void converterNomeComposto() {
            List<String> res =
CamelCase.converterCamelCase("camelCase");

            List<String> expected = new ArrayList<String>();
```

```
            expected.add("camel");
            expected.add("Case");

            assertEquals(res, expected);
      }

      @Test
      public void getWordsArray() {
            String res[] = CamelCase.getWordsArray("CamelCaseTeste");

            assertEquals(res[0], "camel");
            assertEquals(res[1], "Case");
            assertEquals(res[2], "Teste");
      }

      @Test
      public void separarPorLetraMaiuscula() {
            String res[] =
CamelCase.separarPorLetraMaiuscula("camelCaseTeste");

            assertEquals(res[0], "camel");
            assertEquals(res[1], "Case");
            assertEquals(res[2], "Teste");
      }

      @Test
      public void handleFirstWord() {
            String words[] = {"TesTe"};

            String res[] = CamelCase.handleFirstWord(words);

            assertEquals(res[0], "teste");
      }

      @Test
      public void converterPalavraCaixaAlta() {
            List<String> res = CamelCase.converterCamelCase("CPF");

            List<String> expected = new ArrayList<String>();
            expected.add("CPF");

            assertEquals(res, expected);
      }

      @Test
```

```
	public void converterPalavraCaixaAltaCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("nomeCPF");

		List<String> expected = new ArrayList<String>();
		expected.add("nome");
		expected.add("CPF");

		assertEquals(res, expected);
	}

	@Test
	public void converterPalavraNumeroCPFContribuinte() {
		List<String> res =
CamelCase.converterCamelCase("numeroCPFContribuinte");

		List<String> expected = new ArrayList<String>();
		expected.add("numero");
		expected.add("CPF");
		expected.add("Contribuinte");

		assertEquals(res, expected);
	}

	@Test
	public void converterPalavraRecupera10Primeiros() {
		List<String> res =
CamelCase.converterCamelCase("recupera10Primeiros");

		List<String> expected = new ArrayList<String>();
		expected.add("recupera");
		expected.add("10");
		expected.add("Primeiros");

		assertEquals(res, expected);
	}

	@Test(expected=FirstCharException.class)
	public void primeiroCaractereNumero() {
		CamelCase.converterCamelCase("10Primeiros");
	}
}
```

9.2 - Alterando método para validar primeira letra da primeira palavra.

```java
package camelCase;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> lista = new
ArrayList<String>(Arrays.asList(getWordsArray(original)));

		if(String.valueOf(lista.get(0).charAt(0)).matches("[0-9]"))
{
			throw new FirstCharException("O primeiro caractere não
pode ser numerico");
		}

		return lista;
	}


	public static String[] handleFirstWord(String[] words) {
		if (words[0].toUpperCase() != words[0]) {
			words[0] = words[0].toLowerCase();
		}

		return words;
	}

	public static String[] getWordsArray(String original) {
		String words[] = separarPorLetraMaiuscula(original);

		return handleFirstWord(words);
	}

	public static String[] separarPorLetraMaiuscula(String original) {
		String splited[] =
original.split("(?<=[a-z](?=[A-Z0-10{1,}]))|((?=[A-Z][a-z]))");

		return splited;
	}
}
```

10.1- Refatorando, pois a responsabilidade de testar a primeira palavra deveria estar no

método handleFirstWord. Primeiramente adicionando o teste a este método.

```
package camelCase;

import org.junit.Test;

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

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

public class TestCamelCase {

	@Test
	public void converterCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	public void converterSomentePrimeiraLetraMaiuscula() {
		List<String> res = CamelCase.converterCamelCase("Camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	public void converterNomeComposto() {
		List<String> res =
CamelCase.converterCamelCase("camelCase");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");
		expected.add("Case");

		assertEquals(res, expected);
	}
```

```java
	@Test
	public void getWordsArray() {
		String res[] = CamelCase.getWordsArray("CamelCaseTeste");

		assertEquals(res[0], "camel");
		assertEquals(res[1], "Case");
		assertEquals(res[2], "Teste");
	}

	@Test
	public void separarPorLetraMaiuscula() {
		String res[] =
CamelCase.separarPorLetraMaiuscula("camelCaseTeste");

		assertEquals(res[0], "camel");
		assertEquals(res[1], "Case");
		assertEquals(res[2], "Teste");
	}

	@Test
	public void handleFirstWord() {
		String words[] = {"TesTe"};

		String res[] = CamelCase.handleFirstWord(words);

		assertEquals(res[0], "teste");
	}

	@Test
	public void converterPalavraCaixaAlta() {
		List<String> res = CamelCase.converterCamelCase("CPF");

		List<String> expected = new ArrayList<String>();
		expected.add("CPF");

		assertEquals(res, expected);
	}

	@Test
	public void converterPalavraCaixaAltaCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("nomeCPF");

		List<String> expected = new ArrayList<String>();
		expected.add("nome");
		expected.add("CPF");
```

```
            assertEquals(res, expected);
      }

      @Test
      public void converterPalavraNumeroCPFContribuinte() {
            List<String> res =
CamelCase.converterCamelCase("numeroCPFContribuinte");

            List<String> expected = new ArrayList<String>();
            expected.add("numero");
            expected.add("CPF");
            expected.add("Contribuinte");

            assertEquals(res, expected);
      }

      @Test
      public void converterPalavraRecupera10Primeiros() {
            List<String> res =
CamelCase.converterCamelCase("recupera10Primeiros");

            List<String> expected = new ArrayList<String>();
            expected.add("recupera");
            expected.add("10");
            expected.add("Primeiros");

            assertEquals(res, expected);
      }

      @Test(expected=FirstCharException.class)
      public void primeiroCaractereNumero() {
            String words[] = {"0Tests"};

            CamelCase.handleFirstWord(words);
      }

}
```

10.2 - Tranferindo a responsabilidade para o método especializado, e teste passando.

```
package camelCase;

import java.util.ArrayList;
```

```
import java.util.Arrays;
import java.util.List;

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> lista = new
ArrayList<String>(Arrays.asList(getWordsArray(original)));

		return lista;
	}


	public static String[] handleFirstWord(String[] words) {
		if (words[0].toUpperCase() != words[0]) {
			words[0] = words[0].toLowerCase();
		}

		if(String.valueOf(words[0].charAt(0)).matches("[0-9]")) {
			throw new FirstCharException("O primeiro caractere não
pode ser numerico");
		}

		return words;
	}

	public static String[] getWordsArray(String original) {
		String words[] = separarPorLetraMaiuscula(original);



		return handleFirstWord(words);
	}

	public static String[] separarPorLetraMaiuscula(String original) {
		String splited[] =
original.split("(?<=[a-z](?=[A-Z0-10{1,}]))|((?=[A-Z][a-z]))");

		return splited;
	}
}
```

11.1 - Teste de caracteres especiais, não está passando.

```
package camelCase;
```

```java
import org.junit.Test;

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

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

public class TestCamelCase {

	@Test
	public void converterCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	public void converterSomentePrimeiraLetraMaiuscula() {
		List<String> res = CamelCase.converterCamelCase("Camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	public void converterNomeComposto() {
		List<String> res =
CamelCase.converterCamelCase("camelCase");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");
		expected.add("Case");

		assertEquals(res, expected);
	}

	@Test
	public void getWordsArray() {
		String res[] = CamelCase.getWordsArray("CamelCaseTeste");
```

```
            assertEquals(res[0], "camel");
            assertEquals(res[1], "Case");
            assertEquals(res[2], "Teste");
      }

      @Test
      public void separarPorLetraMaiuscula() {
            String res[] =
CamelCase.separarPorLetraMaiuscula("camelCaseTeste");

            assertEquals(res[0], "camel");
            assertEquals(res[1], "Case");
            assertEquals(res[2], "Teste");
      }

      @Test
      public void handleFirstWord() {
            String words[] = {"TesTe"};

            String res[] = CamelCase.handleFirstWord(words);

            assertEquals(res[0], "teste");
      }

      @Test
      public void converterPalavraCaixaAlta() {
            List<String> res = CamelCase.converterCamelCase("CPF");

            List<String> expected = new ArrayList<String>();
            expected.add("CPF");

            assertEquals(res, expected);
      }

      @Test
      public void converterPalavraCaixaAltaCaixaBaixa() {
            List<String> res = CamelCase.converterCamelCase("nomeCPF");

            List<String> expected = new ArrayList<String>();
            expected.add("nome");
            expected.add("CPF");

            assertEquals(res, expected);
      }
```

```
	@Test
	public void converterPalavraNumeroCPFContribuinte() {
		List<String> res =
CamelCase.converterCamelCase("numeroCPFContribuinte");

		List<String> expected = new ArrayList<String>();
		expected.add("numero");
		expected.add("CPF");
		expected.add("Contribuinte");

		assertEquals(res, expected);
	}

	@Test
	public void converterPalavraRecupera10Primeiros() {
		List<String> res =
CamelCase.converterCamelCase("recupera10Primeiros");

		List<String> expected = new ArrayList<String>();
		expected.add("recupera");
		expected.add("10");
		expected.add("Primeiros");

		assertEquals(res, expected);
	}

	@Test(expected=FirstCharException.class)
	public void primeiroCaractereNumero() {
		String words[] = {"0Tests"};

		CamelCase.handleFirstWord(words);
	}

	@Test(expected=SpecialCharException.class)
	public void contemCaracteresEspeciais() {
		CamelCase.converterCamelCase("Tests0Legais&top#xama");
	}

}
```

11.2-Inserindo validação de caracteres especiais no método converterCamelCase

```
package camelCase;
```

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

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> lista = new
ArrayList<String>(Arrays.asList(getWordsArray(original)));

		for (String word : lista) {
			if (word.matches(".*(?![a-zA-Z]|[0-9])(?!$).*")) {
				throw new SpecialCharException("Não sao aceitos
caracteres especiais");
			}
		}

		return lista;
	}

	public static String[] handleFirstWord(String[] words) {
		if (words[0].toUpperCase() != words[0]) {
			words[0] = words[0].toLowerCase();
		}

		if(String.valueOf(words[0].charAt(0)).matches("[0-9]")) {
			throw new FirstCharException("O primeiro caractere não
pode ser numerico");
		}

		return words;
	}

	public static String[] getWordsArray(String original) {
		String words[] = separarPorLetraMaiuscula(original);

		return handleFirstWord(words);
	}

	public static String[] separarPorLetraMaiuscula(String original) {
		String splited[] =
original.split("(?<=[a-z](?=[A-Z0-10{1,}]))|((?=[A-Z][a-z]))");

		return splited;
	}
```

```
}
```

12.1-Refatorando, pois a responsabilidade de testar a primeira palavra deveria estar no método handleFirstWord. Primeiramente adicionando o teste a este método.

```
package camelCase;

import org.junit.Test;

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

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

public class TestCamelCase {

	@Test
	public void converterCaixaBaixa() {
		List<String> res = CamelCase.converterCamelCase("camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	public void converterSomentePrimeiraLetraMaiuscula() {
		List<String> res = CamelCase.converterCamelCase("Camel");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");

		assertEquals(res, expected);
	}

	@Test
	public void converterNomeComposto() {
		List<String> res =
CamelCase.converterCamelCase("camelCase");

		List<String> expected = new ArrayList<String>();
		expected.add("camel");
		expected.add("Case");
```

```
            assertEquals(res, expected);
      }

      @Test
      public void getWordsArray() {
            String res[] = CamelCase.getWordsArray("CamelCaseTeste");

            assertEquals(res[0], "camel");
            assertEquals(res[1], "Case");
            assertEquals(res[2], "Teste");
      }

      @Test
      public void separarPorLetraMaiuscula() {
            String res[] =
CamelCase.separarPorLetraMaiuscula("camelCaseTeste");

            assertEquals(res[0], "camel");
            assertEquals(res[1], "Case");
            assertEquals(res[2], "Teste");
      }

      @Test
      public void handleFirstWord() {
            String words[] = {"TesTe"};

            String res[] = CamelCase.handleFirstWord(words);

            assertEquals(res[0], "teste");
      }

      @Test
      public void converterPalavraCaixaAlta() {
            List<String> res = CamelCase.converterCamelCase("CPF");

            List<String> expected = new ArrayList<String>();
            expected.add("CPF");

            assertEquals(res, expected);
      }

      @Test
      public void converterPalavraCaixaAltaCaixaBaixa() {
            List<String> res = CamelCase.converterCamelCase("nomeCPF");
```

```
            List<String> expected = new ArrayList<String>();
            expected.add("nome");
            expected.add("CPF");

            assertEquals(res, expected);
      }

      @Test
      public void converterPalavraNumeroCPFContribuinte() {
            List<String> res =
CamelCase.converterCamelCase("numeroCPFContribuinte");

            List<String> expected = new ArrayList<String>();
            expected.add("numero");
            expected.add("CPF");
            expected.add("Contribuinte");

            assertEquals(res, expected);
      }

      @Test
      public void converterPalavraRecupera10Primeiros() {
            List<String> res =
CamelCase.converterCamelCase("recupera10Primeiros");

            List<String> expected = new ArrayList<String>();
            expected.add("recupera");
            expected.add("10");
            expected.add("Primeiros");

            assertEquals(res, expected);
      }

      @Test(expected=FirstCharException.class)
      public void primeiroCaractereNumero() {
            String words[] = {"0Tests"};

            CamelCase.handleFirstWord(words);
      }

      @Test(expected=SpecialCharException.class)
      public void contemCaracteresEspeciais() {
            CamelCase.getWordsArray("Tests0Legais&top#xama");
      }
```

```
}
```

12.2- Implementando método throwsIfNotValid em getWordsArray.

```
package camelCase;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> lista = new
ArrayList<String>(Arrays.asList(getWordsArray(original)));

		return lista;
	}

	public static void throwsIfNotValid(String[] words) {
		for (String word : words) {
			if (word.matches(".*(?![a-zA-Z]|[0-9])(?!$).*")) {
				throw new SpecialCharException("Não sao aceitos
caracteres especiais");
			}
		  }
	}

	public static void throwsIfNumericFirstChar(String word) {
		if(String.valueOf(word.charAt(0)).matches("[0-9]")) {
			throw new FirstCharException("O primeiro caractere não
pode ser numerico");
		}
	}

	public static String[] handleFirstWord(String[] words) {
		if (words[0].toUpperCase() != words[0]) {
			words[0] = words[0].toLowerCase();
		}

		throwsIfNumericFirstChar(words[0]);

		return words;
```

```
	}

	public static String[] getWordsArray(String original) {
		String words[] = separarPorLetraMaiuscula(original);

		throwsIfNotValid(words);

		return handleFirstWord(words);
	}

	public static String[] separarPorLetraMaiuscula(String original) {
		String splited[] =
original.split("(?<=[a-z](?=[A-Z0-10{1,}]))|((?=[A-Z][a-z]))");

		return splited;
	}
}
```