Maven foi utilizado para o desenvolvimento deste projeto, por isso essa hierarquia de pacotes. O resultado final não foi alterado por conta disto.

1º CICLO

Teste adicionado (FALHOU)

```
private List<String> palavras;

@Test
public void apenasUmaPalavra() {

	CamelCaseToStringList conversor = new CamelCaseToStringList();
	palavras = conversor.converterCamelCase ("nome");

	assertEquals("nome", convertido);
}
```

Ainda não existia a classe CamelCaseToStringList e nem o método converterCamelCase. Classe e método foram criados e o teste passou

```
public class CamelCaseToStringList {

	private List<String> palavras;

	public List<String> converterCamelCase(String palavra) {
		palavras = new ArrayList<String>();

		palavras.add(palavra);

		return palavras;
	}
}
```

2º CICLO

Teste adicionado (FALHOU)

```
@Test
public void converteDuasPalavras() {

	palavras = conversor.converterCamelCase("nomeComposto");

	assertEquals("nome", palavras.get(0));
	assertEquals("composto", palavras.get(1));
}
```

Elaborada lógica para pegar letra maiúscula. O teste passou

```
public class CamelCaseToStringList {

	private List<String> palavras;
	private int index = 0;

	public List<String> converterCamelCase(String original) {
		palavras = new ArrayList<String>();

		for (int i = 0; i < original.length(); i++) {
			if (original.length() -1 == i) {
				palavras.add(original.substring(index, i +1).toLowerCase());
			} else if (Character.isUpperCase(original.charAt(i))) {
				palavras.add(original.substring(index, i).toLowerCase());
				index = i;
			}
		}

		return palavras;
	}
}
```

3º CICLO

Teste Adicionado (FALHOU)

```
@Test
public void converteUmaPalavraComPrimeiraLetraMaiuscula() {

	palavras = conversor.converterCamelCase("Nome");

	assertEquals("nome", palavras.get(0));
}
```

Dentro do for foi colocado int i = 1 para desconsiderar a 1ª letra. O teste passou.

```
public List<String> converterCamelCase(String original) {
	palavras = new ArrayList<String>();

	for (int i = 1; i < original.length(); i++) {
		if (original.length() -1 == i) {
			palavras.add(original.substring(index, i +1).toLowerCase());
		} else if (Character.isUpperCase(original.charAt(i))) {
			palavras.add(original.substring(index, i).toLowerCase());
			index = i;
		}
	}
	if (palavras.isEmpty()) {
		palavras.add(original);
	}
	return palavras;
}
```

4º CICLO

Teste Adicionado (PASSOU). Deixado apenas para manter essa validação

```
@Test
public void converteDuasPalavrasComPrimeiraLetraMaiuscula() {

	palavras = conversor.converterCamelCase("NomeComposto");

	assertEquals("nome", palavras.get(0));
	assertEquals("composto", palavras.get(1));
}
```

Teste Adicionado (PASSOU). Deixado apenas para manter essa validação

```
@Test
public void converteSeisPalavras() {

	palavras = conversor.converterCamelCase("QuebraDeStringsComCamelCase");

	assertEquals("quebra", palavras.get(0));
	assertEquals("de", palavras.get(1));
	assertEquals("strings", palavras.get(2));
	assertEquals("com", palavras.get(3));
	assertEquals("camel", palavras.get(4));
	assertEquals("case", palavras.get(5));
}
```

Teste Adicionado (FALHOU)

```
@Test
public void todasMaiusculas() {

	palavras = conversor.converterCamelCase("CPF");

	assertEquals("CPF", palavras.get(0));
}
```

Adicionado novo for para verificar se todas as letras são maiúsculas e cercado for antigo dentro de um if para só rodar caso exista alguma minúscula.

```
private List<String> palavras;
private int index = 0;
private boolean todasSaoMaiusculas = true;

public List<String> converterCamelCase(String original) {
	palavras = new ArrayList<String>();

	for (int i = 0; i < original.length(); i++) {
		if (Character.isLowerCase(original.charAt(i))){
			todasSaoMaiusculas = false;
		}
	}

	if (todasSaoMaiusculas){
		palavras.add(original);
	} else {
		for (int i = 1; i < original.length(); i++) {
			if (original.length() -1 == i) {
				palavras.add(original.substring(index, i +1).toLowerCase());
			} else if (Character.isUpperCase(original.charAt(i))) {
				palavras.add(original.substring(index, i).toLowerCase());
				index = i;
			}
		}
	}

	if (palavras.isEmpty()) {
		palavras.add(original);
	}

	return palavras;
}
```

5º CICLO

Teste Adicionado (FALHOU)

```
@Test(expected=CaracterEspecialException.class)
public void converteCaracterEspecial() throws CaracterEspecialException {

	palavras = conversor.converterCamelCase("nome#Composto");
}
```

Criada classe de exceção e criado método para validar se existe caractere especial na palavra original. Lança uma exceção caso exista. O teste passou

```
private void verificarCaracterEspecial(String original) throws CaracterEspecialException {
	for (int i = 0; i < original.length(); i++) {
		if (!Character.isLetterOrDigit(original.charAt(i))){
			throw new CaracterEspecialException("Inválido caracteres especiais não são
permitidos, somente letras e números");
		}
	}
}
```

6º CICLO

Adicionado Teste (FALHOU)

```
@Test(expected=StringVaziaException.class)
public void stringVazia() throws CaracterEspecialException, StringVaziaException{
	palavras = CamelCaseToStringList.converterCamelCase("");
}
```

Criada classe de exception e método para lançar a exceção caso de o erro.

```
private static void verificaStringVazia(String original) throws StringVaziaException {
	if (original.isEmpty())
		throw new StringVaziaException("Inválido - a string está vazia");
}
```

O Teste Passou

7º CICLO

Teste Adicionado (FALHOU)

```
@Test
public void converteMaiusculasEMinusculas() throws CaracterEspecialException {

	palavras = conversor.converterCamelCase("numeroCPF");

	assertEquals("numero", palavras.get(0));
	assertEquals("CPF", palavras.get(1));
}
```

O código sofreu diversas alterações até na lógica correta.

```
public static List<String> converterCamelCase(String original) throws
CaracterEspecialException, StringVaziaException {
	verificaCaracterEspecial(original);
	verificaStringVazia(original);

	List<String> palavras = new ArrayList<>();
	int index = 0;
	char anterior;
	String palavraMaiuscula = "";

	for (int i = 1; i < original.length(); i++) {

		anterior = original.charAt(i -1);

		if (Character.isUpperCase(anterior) && Character.isUpperCase(original.charAt(i)))
{


			if (palavraMaiuscula.length() == 0) {

				palavraMaiuscula = palavraMaiuscula + anterior;
				palavraMaiuscula = palavraMaiuscula + original.charAt(i);

			} else {

				palavraMaiuscula = palavraMaiuscula + original.charAt(i);

			}

			if (original.length() -1 == i) {

				palavras.add(palavraMaiuscula);
				palavraMaiuscula = "";
			}


		} else if (Character.isUpperCase(anterior) &&
!Character.isUpperCase(original.charAt(i)) && i > 1 && palavraMaiuscula.length() > 0) {

			palavras.add(palavraMaiuscula.substring(0, palavraMaiuscula.length() -1));
			palavraMaiuscula = "";

		} else if (original.length() -1 == i) {

			palavras.add(original.substring(index, i +1).toLowerCase());
```

```
            } else if (Character.isUpperCase(original.charAt(i))) {

                  palavras.add(original.substring(index, i).toLowerCase());
                  index = i;

            }
      }

      if (palavras.isEmpty()) {
            palavras.add(original.toLowerCase());
      }
      return palavras;
}
```

8º CICLO

Adicionado Teste (FALHOU)

```
@Test
public void converteMaiusculasEMinusculasTeste2() throws CaracterEspecialException,
StringVaziaException {

	palavras = CamelCaseToStringList.converterCamelCase("numeroCPFContribuinte");

	assertEquals("numero", palavras.get(0));
	assertEquals("CPF", palavras.get(1));
	assertEquals("contribuinte", palavras.get(2));
}
```

Para que o teste passasse, foi adicionado apenas index = i -1 no if abaixo:

```
} else if (Character.isUpperCase(anterior) && !Character.isUpperCase(original.charAt(i)) && i >
1 && palavraMaiuscula.length() > 0) {

	palavras.add(palavraMaiuscula.substring(0, palavraMaiuscula.length() -1));
	palavraMaiuscula = "";

	index = i -1;

}
```

Os testes passaram!

9º CICLO

Adicionado Teste (FALHOU)

```
@Test
public void convertePalavraComNumeros() throws CaracterEspecialException, StringVaziaException
{

	palavras = CamelCaseToStringList.converterCamelCase("recupera10Primeiros ");

	assertEquals("recupera", palavras.get(0));
	assertEquals("10", palavras.get(1));
	assertEquals("primeiros", palavras.get(2));
}
```

Foi adicionado mais clausulas if dentro do for para cobrir esse cenário com número. O teste passou.

Abaixo está marcado o trecho adicionado

```
for (int i = 1; i < original.length(); i++) {

	anterior = original.charAt(i -1);

	if (Character.isUpperCase(anterior) && Character.isUpperCase(original.charAt(i))) {

		if (palavraMaiuscula.length() == 0) {

			palavraMaiuscula = palavraMaiuscula + anterior;
			palavraMaiuscula = palavraMaiuscula + original.charAt(i);

		} else {

			palavraMaiuscula = palavraMaiuscula + original.charAt(i);

		}

		if (original.length() -1 == i) {

			palavras.add(palavraMaiuscula);
			palavraMaiuscula = "";

		}

	} else if (Character.isUpperCase(anterior) && !Character.isUpperCase(original.charAt(i))
&& i > 1 && palavraMaiuscula.length() > 0) {

		palavras.add(palavraMaiuscula.substring(0, palavraMaiuscula.length() -1));
		palavraMaiuscula = "";
		index = i -1;

	} else if (original.length() -1 == i) {

		palavras.add(original.substring(index, i +1).toLowerCase());

	} else if (Character.isUpperCase(original.charAt(i))) {

		palavras.add(original.substring(index, i).toLowerCase());
		index = i;

	} else if (Character.isDigit(original.charAt(i))) {
```

```
            if (numero.isEmpty()){
                  palavras.add(original.substring(index, i).toLowerCase());
                  index = i;
            }
            numero = numero + original.charAt(i);

      } else if (!numero.isEmpty() && Character.isDigit(original.charAt(i -1))) {

            palavras.add(numero);
            numero = "";

      }

}
```

10º CICLO

Adicionado o teste (FALHOU)

```
@Test(expected=IniciaComNumeroException.class)
public void convertePalavraIniciandoComNumeros() throws CaracterEspecialException,
StringVaziaException, IniciaComNumeroException {

	palavras = CamelCaseToStringList.converterCamelCase("10Primeiros");
}
```

Criada classe de exception e método que lança essa exceção caso a palavra seja iniciada com números.

```
private static void verificaSeIniciaComNumero(String original) throws IniciaComNumeroException
{
	if (Character.isDigit(original.charAt(0)))
		throw new IniciaComNumeroException("Inválido - não deve começar com números");
}
```

11º CICLO

Refatoração.

Nesse momento a classe CamelCaseToStringList está assim:

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

public class CamelCaseToStringList {

	public static List<String> converterCamelCase(String original) throws
CaracterEspecialException, StringVaziaException, IniciaComNumeroException {
		verificaStringVazia(original);
		verificaCaracterEspecial(original);
		verificaSeIniciaComNumero(original);

		List<String> palavras = new ArrayList<>();
		int index = 0;
		char anterior;
		String palavraMaiuscula = "";
		String numero = "";

		for (int i = 1; i < original.length(); i++) {

			anterior = original.charAt(i -1);

			if (Character.isUpperCase(anterior) &&
Character.isUpperCase(original.charAt(i))) {

				if (palavraMaiuscula.length() == 0) {

					palavraMaiuscula = palavraMaiuscula + anterior;
					palavraMaiuscula = palavraMaiuscula + original.charAt(i);

				} else {

					palavraMaiuscula = palavraMaiuscula + original.charAt(i);

				}

				if (original.length() -1 == i) {

					palavras.add(palavraMaiuscula);
					palavraMaiuscula = "";

				}

			} else if (Character.isUpperCase(anterior) &&
!Character.isUpperCase(original.charAt(i)) && i > 1 && palavraMaiuscula.length() > 0) {

				palavras.add(palavraMaiuscula.substring(0, palavraMaiuscula.length()
-1));
				palavraMaiuscula = "";
				index = i -1;

			} else if (original.length() -1 == i) {

				palavras.add(original.substring(index, i +1).toLowerCase());

			} else if (Character.isUpperCase(original.charAt(i))) {

				palavras.add(original.substring(index, i).toLowerCase());
				index = i;
```

```
                } else if (Character.isDigit(original.charAt(i))) {

                        if (numero.isEmpty()){
                                palavras.add(original.substring(index, i).toLowerCase());
                                index = i;
                        }
                        numero = numero + original.charAt(i);

                } else if (!numero.isEmpty() && Character.isDigit(original.charAt(i -1))) {

                        palavras.add(numero);
                        numero = "";

                }

        }

        if (palavras.isEmpty()) {
                palavras.add(original.toLowerCase());
        }

        return palavras;
    }

    private static void verificaCaracterEspecial(String original) throws
CaracterEspecialException {
        for (int i = 0; i < original.length(); i++) {
                if (!Character.isLetterOrDigit(original.charAt(i)))
                        throw new CaracterEspecialException("Inválido - caracteres especiais
não são permitidos, somente letras e números");
        }
    }

    private static void verificaStringVazia(String original) throws StringVaziaException {
        if (original.isEmpty())
                throw new StringVaziaException("Inválido - a string está vazia");
    }

    private static void verificaSeIniciaComNumero(String original) throws
IniciaComNumeroException {
        if (Character.isDigit(original.charAt(0)))
                throw new IniciaComNumeroException("Inválido - não deve começar com
números");
    }

}
```

Na refatoração foram criados diversos métodos para abstrair algumas linhas de código do método principal, inclusive métodos de controle das variáveis. Todos os testes continuaram passando após toda a refatoração.

CamelCaseToStringList após refatoração:

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

public class CamelCaseToStringList {

    private static List<String> palavras = new ArrayList<>();
    private static int index = 0;
    private static char anterior;
    private static String palavraMaiuscula = "";
    private static String numero = "";
```

```
	private CamelCaseToStringList() {
		super();
	}

	private static void adicionaNaListaDePalavras(String palavra) {
		palavras.add(palavra);
	}

	public static void limparListaDePalavras() {
		palavras.clear();
	}

	public static void setIndex(int i) {
		index = i;
	}

	public static void zerarIndex() {
		index = 0;
	}

	public static void setAnterior(char c) {
		anterior = c;
	}

	public static void limparPalavraMaiuscula() {
		palavraMaiuscula = "";
	}

	public static void setPalavraMaiuscula(String s) {
		palavraMaiuscula = s;
	}

	public static void limparNumero() {
		 numero = "";
	}

	public static void setNumero(String s) {
		numero = s;
	}

	public static List<String> converterCamelCase(String original) throws ErroDeValidacao {
		validacaoInicial(original);
		for (int i = 1; i < original.length(); i++) {
			setAnterior(original.charAt(i - 1));
			efetuaValidacoesEPopulaListaDePalavras(original, i);
		}
		if (palavras.isEmpty())
			adicionaNaListaDePalavras(original.toLowerCase());
		return palavras;
	}

	private static void efetuaValidacoesEPopulaListaDePalavras(String original, int i) {
		if (caracterAnteriorEAtualSaoMaiusculos(original, anterior, i))
			setPalavraMaiuscula(validacoesParaPalavraMaiuscula(original, anterior,
palavraMaiuscula, i));
		else if (palavraMaiusculaDeveSerAdicionadaAlista(original, anterior,
palavraMaiuscula, i))
			adicionaEAtualizaIndexELimpaPalavraMaiuscula(i);
		else if (original.length() - 1 == i)
			adicionaNaListaDePalavras(original.substring(index, i + 1).toLowerCase());
		else if (Character.isUpperCase(original.charAt(i)))
			adicionaNaListaDePalavrasEAtualizaIndex(original, i);
		else if (Character.isDigit(original.charAt(i)))
			efetuaMovimentacaoParaNumero(original, i);
		else if (numeroNaoEstaVazioECharAnteriorEUmDigito(original, i))
```

```
                adicionaNaListaDePalavrasELimpaNumero();
        }

        private static void adicionaNaListaDePalavrasELimpaNumero() {
                adicionaNaListaDePalavras(numero);
                limparNumero();
        }

        private static void efetuaMovimentacaoParaNumero(String original, int i) {
                if (numero.isEmpty())
                        adicionaNaListaDePalavrasEAtualizaIndex(original, i);
                setNumero(numero + original.charAt(i));
        }

        private static boolean numeroNaoEstaVazioECharAnteriorEUmDigito(String original, int i)
{
                return !numero.isEmpty() && Character.isDigit(original.charAt(i - 1));
        }

        private static void adicionaNaListaDePalavrasEAtualizaIndex(String original, int i) {
                adicionaNaListaDePalavras(original.substring(index, i).toLowerCase());
                setIndex(i);
        }

        private static void adicionaEAtualizaIndexELimpaPalavraMaiuscula(int i) {
                adicionaNaListaDePalavras(palavraMaiuscula.substring(0, palavraMaiuscula.length()
- 1));
                limparPalavraMaiuscula();
                setIndex(i - 1);
        }

        private static boolean caracterAnteriorEAtualSaoMaiusculos(String original, char
anterior, int i) {
                return Character.isUpperCase(anterior) &&
Character.isUpperCase(original.charAt(i));
        }

        private static String validacoesParaPalavraMaiuscula(String original, char anterior,
String palavraMaiuscula, int i) {
                String ret;
                if (palavraMaiuscula.length() == 0)
                        ret = palavraMaiuscula + anterior + original.charAt(i);
                else
                        ret = palavraMaiuscula + original.charAt(i);
                if (original.length() - 1 == i)
                        adicionaNaListaDePalavrasELimpaPalavraMaiuscula(ret);
                return ret;
        }

        private static void adicionaNaListaDePalavrasELimpaPalavraMaiuscula(String
palavraMaiuscula) {
                adicionaNaListaDePalavras(palavraMaiuscula);
                limparPalavraMaiuscula();
        }

        private static void validacaoInicial(String original) throws StringVaziaException,
CaracterEspecialException, IniciaComNumeroException {
                verificaStringVazia(original);
                verificaCaracterEspecial(original);
                verificaSeIniciaComNumero(original);
        }

        private static boolean palavraMaiusculaDeveSerAdicionadaALista(String original, char
anterior, String palavraMaiuscula, int i) {
```

```
            return Character.isUpperCase(anterior) &&
!Character.isUpperCase(original.charAt(i)) && i > 1 && palavraMaiuscula.length() > 0;
      }

      private static void verificaCaracterEspecial(String original) throws
CaracterEspecialException {
            for (int i = 0; i < original.length(); i++) {
                  if (!Character.isLetterOrDigit(original.charAt(i)))
                        throw new CaracterEspecialException( "Inválido - caracteres especiais
não são permitidos, somente letras e números");
            }
      }

      private static void verificaStringVazia(String original) throws StringVaziaException {
            if (original.isEmpty())
                  throw new StringVaziaException("Inválido - a string está vazia");
      }

      private static void verificaSeIniciaComNumero(String original) throws
IniciaComNumeroException {
            if (Character.isDigit(original.charAt(0)))
                  throw new IniciaComNumeroException("Inválido - não deve começar com
números");
      }

}
```