Primeiro Ciclo TDD:

```java
@Test
public void testUmaPalavra() {
    List<String> resultados = CamelCase.converteCamelCase("nome");
    assertEquals("nome", resultados.get(0));
}
```

Código fonte antes: Não existia

Código fonte depois:

```java
public class CamelCase {

    static List<String> resultados = new ArrayList<String>();

    public static List<String> converteCamelCase(String original) {
        resultados.add(original);
        return resultados;
    }

}
```

Segundo Ciclo TDD:

```
    List<String> resultados;

    @After
    public void limpaArrayCamelCase() {
        resultados.clear();
    }

    @Test
    public void testUmaPalavra() {
        resultados = CamelCase.converteCamelCase("nome");
        assertEquals("nome", resultados.get(0));
    }

    @Test
    public void testUmaPalavraInicialMaiuscula() {
        resultados = CamelCase.converteCamelCase("Nome");
        assertEquals("nome", resultados.get(0));
    }
```

Código fonte antes: Referente ao 1º Ciclo

Código fonte depois:

```
    static List<String> resultados = new ArrayList<String>();

    public static List<String> converteCamelCase(String original) {
        resultados.add(original.toLowerCase());
        return resultados;
    }
```

O que foi feito: Foi adicionado o na variável original o método toLowerCase()

Terceiro Ciclo TDD:

```java
@Test
public void testNomeComposto() {
    resultados = CamelCase.converteCamelCase("nomeComposto");
    assertEquals("nome", resultados.get(0));
    assertEquals("composto", resultados.get(1));
}
```

Código fonte antes: Referente ao 2º Ciclo

Código fonte depois:

```java
public static List<String> converteCamelCase(String original) {
    String palavraFormada = "";
    for (int i = 0; i < original.length(); i++) {
        char letra = original.charAt(i);
        if (isUpperCase(letra) && !isPrimeiraLetra(i)) {
            resultados.add(palavraFormada);
            palavraFormada = "";
        }
        palavraFormada += letra;
    }
    resultados.add(palavraFormada.toLowerCase());
    return resultados;
}

private static boolean isPrimeiraLetra(int i) {
    return i == 0;
}

public static boolean isUpperCase(char letra) {
    return Character.toString(letra).matches("[A-Z]");
}
```

O que foi feito: Fiz uma lógica para formar palavras, e alguns métodos auxiliares: isUpperCase, isPrimeiraLetra. Isso para formar as palavras de maneira correta e colocar dentro da lista de resultados.

Quarto Ciclo TDD:

```
@Test
public void testNomeCompostoComNumeros(){
    resultados = CamelCase.converteCamelCase("recupera10Primeiros");
    assertEquals("recupera", resultados.get(0));
    assertEquals("10", resultados.get(1));
    assertEquals("primeiros", resultados.get(2));
}
```

Código fonte antes: Referente ao 3º Ciclo

Código fonte depois:

```
public static List<String> converteCamelCase(String original) {
    separadorDePalavras(original);
    return resultados;
}

private static void separadorDePalavras(String original) {
    String palavraFormada = "";
    for (int i = 0; i < original.length(); i++) {
        char letra = original.charAt(i);
        palavraFormada = trataEntradaDeNumero(original, palavraFormada, i);
        if (isUpperCase(letra) && !isPrimeiraLetra(i)) {
            resultados.add(palavraFormada);
            palavraFormada = "";
        }
        palavraFormada += letra;
    }
    resultados.add(palavraFormada.toLowerCase());
}

private static String trataEntradaDeNumero(String original, String palavraFormada, int i) {
    if (Character.isDigit(original.charAt(i)) && !Character.isDigit(original.charAt(i - 1))) {
        resultados.add(palavraFormada.toLowerCase());
        palavraFormada = "";
    }
    return palavraFormada;
}
```

O que foi feito: Fiz uma lógica para quando tiver números em meio as palavras. Dentro do método sepadorDePalavras() tem uma chamado ao método trataEntradaDeNumero()

Quinto Ciclo TDD:

```
@Test(expected=ComecaComNumeroException.class)
public void testComecaComNumero() {
    resultados = CamelCase.converteCamelCase("10Primeiros");
    fail();
    assertEquals("10", resultados.get(0));
    assertEquals("primeiros", resultados.get(1));
}
```

Código fonte antes: Referente ao 4º Ciclo

Código fonte depois:

```
private static final String COMECA_COM_NUMERO_EXCEPTION = "Palavra não pode começar com um número";
static List<String> resultados = new ArrayList<String>();

public static List<String> converteCamelCase(String original) {
    comecaComNumero(original);
    separadorDePalavras(original);

    return resultados;
}

private static void comecaComNumero(String original) {
    String primeiraLetra = Character.toString(original.charAt(0));
    if(primeiraLetra.matches("[0-9]"))
        throw new ComecaComNumeroException(COMECA_COM_NUMERO_EXCEPTION);
}
```

Sexto Ciclo TDD:

```
@Test(expected = CharacterInvalidoException.class)
public void testCaractereInvalido() {
    resultados = CamelCase.converteCamelCase("nome#Composto");
    fail();
    assertEquals("nome", resultados.get(0));
}
```

Código fonte antes: Referente ao 5º Ciclo

Código fonte depois:

```
private static final String COMECA_COM_NUMERO_EXCEPTION = "Palavra não pode começar com um número";
private static final String CHARACTER_EXCEPTION = "Palavra não pode conter Characteres especiais";
static List<String> resultados = new ArrayList<String>();

public static List<String> converteCamelCase(String original) {
    comecaComNumero(original);
    caractereInvalido(original);
    separadorDePalavras(original);

    return resultados;
}

 private static void caractereInvalido(String original) {
     boolean validacao = original.matches(".*\\W.*");
     if (validacao) {
         throw new CharacterInvalidoException(CHARACTER_EXCEPTION);
     }
 }
```

Sétimo Ciclo TDD:

```
@Test(expected = PalavraVaziaException.class)
public void testPalavraVazia() {
    resultados = CamelCase.converteCamelCase("");
    fail();
    assertEquals("", resultados.get(0));
}
```

Código fonte antes: Referente ao 5º Ciclo

Código fonte depois:

```
private static final String COMECA_COM_NUMERO_EXCEPTION = "Palavra não pode começar com um número";
private static final String CHARACTER_EXCEPTION = "Palavra não pode conter Characteres especiais";
private static final String PALAVRA_VAZIA_EXCEPTION = "Palavra não pode ser vazia";
static List<String> resultados = new ArrayList<String>();

public static List<String> converteCamelCase(String original) {
    palavraVazia(original);
    comecaComNumero(original);
    caractereInvalido(original);
    separadorDePalavras(original);

    return resultados;
}

private static void palavraVazia(String original) {
    if (original.isEmpty()) {
        throw new PalavraVaziaException(PALAVRA_VAZIA_EXCEPTION);
    }
}
```