Semana 1 - Camel Case

Ciclo: 1

Teste Adicionado

```
@Test
public void testSingleWordLowerCase() {
    String camelCaseString = "nome";
    List<String> wordList = CamelCase.converterCamelCase(camelCaseString);
    assertThat(wordList, equalTo(Arrays.asList("nome")));
}
```

Código Anterior

No primeiro ciclo não existe código anterior

Código Novo

```
static List<String> converterCamelCase(String camelCaseString) {
    return Arrays.asList(camelCaseString);
}
```

Descrição

Foi criada uma classe com o método especificado que retorna uma lista com a palavra passada

Ciclo: 2

Teste Adicionado

```
@Test
public void testSingleCaptalizedWord() {
    String camelCaseString = "Nome";
    List<String> wordList = CamelCase.converterCamelCase(camelCaseString);
    assertThat(wordList, equalTo(Arrays.asList("nome")));
}
```

Código Anterior

```
static List<String> converterCamelCase(String camelCaseString) {
    return Arrays.asList(camelCaseString);
}
```

Código Novo

```
static List<String> converterCamelCase(String camelCaseString) {
    return Arrays.asList(camelCaseString.toLowerCase());
}
```

Descriçao

Adicionada transformação para lower case.

Ciclo: 3

Teste Adicionado

```
@Test
public void testComposedWord() {
    String camelCaseString = "nomeComposto";
    List<String> wordList = CamelCase.converterCamelCase(camelCaseString);

    assertThat(wordList, equalTo(Arrays.asList("nome", "composto")));
}
```

Código Anterior

```
static List<String> converterCamelCase(String camelCaseString) {
    return Arrays.asList(camelCaseString.toLowerCase());
}
```

Código Novo

```
static List<String> converterCamelCase(String camelCaseString) {
    List<String> words = new ArrayList<>();

    int wordStart=0;
    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if ((camelCaseString.toCharArray()[i] >= 'A' &&
                camelCaseString.toCharArray()[i] <= 'Z') &&
                i != wordStart) {

            words.add(camelCaseString.substring(wordStart, i).toLowerCase());
            wordStart = i;
        }
    }
    words.add(camelCaseString.substring(wordStart,
camelCaseString.length()).toLowerCase());
    return words;
}
```

Descriçao

Método simplificado não é mais capaz de resolver a separação de palavras. Foi adicionado algoritmo para a separação de palavras ao encontrar letras maiúsculas.

Ciclo: 4

Teste Adicionado

```
@Test
    public void testCaptalizedComposedWord() {
        String camelCaseString = "NomeComposto";
        List<String> wordList = CamelCase.converterCamelCase(camelCaseString);

        assertThat(wordList, equalTo(Arrays.asList("nome", "composto")));
    }

    @Test
    public void testAcronym() {
        String camelCaseString = "CPF";
        List<String> wordList = CamelCase.converterCamelCase(camelCaseString);

        assertThat(wordList, equalTo(Arrays.asList("CPF")));
    }
```

Código Anterior

```
static List<String> converterCamelCase(String camelCaseString) {
    List<String> words = new ArrayList<>();

    int wordStart=0;
    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if ((camelCaseString.toCharArray()[i] >= 'A' &&
                camelCaseString.toCharArray()[i] <= 'Z') &&
                i != wordStart) {

            words.add(camelCaseString.substring(wordStart, i).toLowerCase());
            wordStart = i;
        }
    }
    words.add(camelCaseString.substring(wordStart,
camelCaseString.length()).toLowerCase());
    return words;
}
```

Código Novo

```
private static final Pattern acronym = Pattern.compile("[A-Z]+");

static List<String> converterCamelCase(String camelCaseString) {
    List<String> words = new ArrayList<>();

    int wordStart = 0;
    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if ((camelCaseString.toCharArray()[i] >= 'A'
                && camelCaseString.toCharArray()[i] <= 'Z')
                && i != wordStart
                && !(camelCaseString.toCharArray()[i - 1] >= 'A'
                && camelCaseString.toCharArray()[i - 1] <= 'Z')) {

            String word = camelCaseString.substring(wordStart, i);
            if (!acronym.matcher(word).matches()) {
                word = word.toLowerCase();
            }
            words.add(word);
            wordStart = i;
        }
    }
    String word = camelCaseString.substring(wordStart, camelCaseString.length());
    if (!acronym.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    words.add(word);
    return words;
}
```

Descriçao

Como o primeiro teste adicionado para o ciclo passou, um novo teste teve de ser adicionado. algumas modificações diveram que ser feitas para considerar o caso de acrônimos.

Refatoração

Código Anterior

```
private static final Pattern acronym = Pattern.compile("[A-Z]+");

static List<String> converterCamelCase(String camelCaseString) {
    List<String> words = new ArrayList<>();

    int wordStart = 0;
    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if ((camelCaseString.toCharArray()[i] >= 'A'
                && camelCaseString.toCharArray()[i] <= 'Z')
                && i != wordStart
                && !(camelCaseString.toCharArray()[i - 1] >= 'A'
                && camelCaseString.toCharArray()[i - 1] <= 'Z')) {

            String word = camelCaseString.substring(wordStart, i);
            if (!acronym.matcher(word).matches()) {
                word = word.toLowerCase();
            }
            words.add(word);
            wordStart = i;
        }
    }
    String word = camelCaseString.substring(wordStart, camelCaseString.length());
    if (!acronym.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    words.add(word);
    return words;
}
```

Código Novo

```
private static final Pattern acronym = Pattern.compile("[A-Z]+");

static List<String> converterCamelCase(String camelCaseString) {
    List<String> words = new ArrayList<>();

    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if (shouldBreak(camelCaseString, i)) {

            words.add(formatWord(camelCaseString.substring(0, i)));
            camelCaseString = camelCaseString.substring(i);
        }
    }

    words.add(formatWord(camelCaseString));
    return words;
}

private static boolean shouldBreak(String camelCaseString, int i) {
    return isCapitalLetter(camelCaseString.toCharArray()[i])
            && i != 0
            && !isCapitalLetter(camelCaseString.toCharArray()[i - 1]);
}

private static boolean isCapitalLetter(char c) {
    return c >= 'A' && c <= 'Z';
}

private static String formatWord(String word) {
    if (!acronym.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    return word;
}
```

Descriçao

Como o método ficou muito longo, uma refatoração precisou ser feita. repetições foram extraídas em métodos e a legibilidade foi melhorada.

Ciclo: 4

Teste Adicionado

```
@Test
public void testComposedWordWithAcronym() {
```

```java
    String camelCaseString = "numeroCPF";
    List<String> wordList = CamelCase.converterCamelCase(camelCaseString);

    assertThat(wordList, equalTo(Arrays.asList("numero", "CPF")));
}

@Test
public void testComposedWordWithAcronymInTheMiddle() {
    String camelCaseString = "numeroCPFContribuinte";
    List<String> wordList = CamelCase.converterCamelCase(camelCaseString);

    assertThat(wordList, equalTo(Arrays.asList("numero", "CPF", "contribuinte")));
}
```

Código Anterior

```java
private static final Pattern acronym = Pattern.compile("[A-Z]+");

static List<String> converterCamelCase(String camelCaseString) {
    List<String> words = new ArrayList<>();

    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if (shouldBreak(camelCaseString, i)) {

            words.add(formatWord(camelCaseString.substring(0, i)));
            camelCaseString = camelCaseString.substring(i);
        }
    }

    words.add(formatWord(camelCaseString));
    return words;
}

private static boolean shouldBreak(String camelCaseString, int i) {
    return isCapitalLetter(camelCaseString.toCharArray()[i])
            && i != 0
            && !isCapitalLetter(camelCaseString.toCharArray()[i - 1]);
}

private static boolean isCapitalLetter(char c) {
    return c >= 'A' && c <= 'Z';
}

private static String formatWord(String word) {
    if (!acronym.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    return word;
}
```

Código Novo

```java
private static final Pattern acronym = Pattern.compile("[A-Z]+");
```

```
static List<String> converterCamelCase(String camelCaseString) {
    List<String> words = new ArrayList<>();

    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if (shouldBreak(camelCaseString, i)) {

            words.add(formatWord(camelCaseString.substring(0, i)));
            camelCaseString = camelCaseString.substring(i);
            i = 0;
        }
    }

    words.add(formatWord(camelCaseString));
    System.out.println(String.join(", ", words));
    return words;
}

private static boolean shouldBreak(String camelCaseString, int i) {
    return isCapitalLetter(camelCaseString.toCharArray()[i])
            && i != 0
            && (!isCapitalLetter(camelCaseString.toCharArray()[i - 1])
            || (i < camelCaseString.length() - 1
            && !isCapitalLetter(camelCaseString.toCharArray()[i + 1])));
}

private static boolean isCapitalLetter(char c) {
    return c >= 'A' && c <= 'Z';
}

private static String formatWord(String word) {
    if (!acronym.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    return word;
}
```

Descriçao

Novamente foram adicionados dois testes. Como o segundo teste possuía 3 palavras um erro no código anteior foi percebido e corrigido.

Ciclo: 5

Teste Adicionado

```
@Test
public void testComposedWordWithNumberInTheMiddle() {
    String camelCaseString = "recupera10Primeiros";
    List<String> wordList = CamelCase.converterCamelCase(camelCaseString);

    assertThat(wordList, equalTo(Arrays.asList("recupera", "10", "primeiros")));
}
```

Código Anterior

```
private static final Pattern acronym = Pattern.compile("[A-Z]+");

static List<String> converterCamelCase(String camelCaseString) {
    List<String> words = new ArrayList<>();

    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if (shouldBreak(camelCaseString, i)) {

            words.add(formatWord(camelCaseString.substring(0, i)));
            camelCaseString = camelCaseString.substring(i);
            i = 0;
        }
    }

    words.add(formatWord(camelCaseString));
    System.out.println(String.join(", ", words));
    return words;
}

private static boolean shouldBreak(String camelCaseString, int i) {
    return isCapitalLetter(camelCaseString.toCharArray()[i])
            && i != 0
            && (!isCapitalLetter(camelCaseString.toCharArray()[i - 1])
            || (i < camelCaseString.length() - 1
            && !isCapitalLetter(camelCaseString.toCharArray()[i + 1])));
}

private static boolean isCapitalLetter(char c) {
    return c >= 'A' && c <= 'Z';
}

private static String formatWord(String word) {
    if (!acronym.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    return word;
}
```

Código Novo

```
private static final Pattern acronym = Pattern.compile("[A-Z]+");

static List<String> converterCamelCase(String camelCaseString) {
    List<String> words = new ArrayList<>();

    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if (shouldBreak(camelCaseString, i)) {

            words.add(formatWord(camelCaseString.substring(0, i)));
            camelCaseString = camelCaseString.substring(i);
            i = 0;
        }
    }
    words.add(formatWord(camelCaseString));
    System.out.println(String.join(", ", words));
    return words;
}

private static boolean shouldBreak(String camelCaseString, int i) {
    return isCapitalLetterOrNumber(camelCaseString.toCharArray()[i])
            && i != 0
            && (!isCapitalLetterOrNumber(camelCaseString.toCharArray()[i - 1])
            || (i < camelCaseString.length() - 1
            && !isCapitalLetterOrNumber(camelCaseString.toCharArray()[i + 1])));
}

private static boolean isCapitalLetterOrNumber(char c) {
    return (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9');
}

private static String formatWord(String word) {
    if (!acronym.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    return word;
}
```

Descriçao

Pequena alteração para incluir números como divisores.

Ciclo: 6

Teste Adicionado

```
@Test(expected = InvalidNumberStartException.class)
public void testInvalidNumberStart() {
    String camelCaseString = "10Primeiros";
    List<String> wordList = CamelCase.converterCamelCase(camelCaseString);
}
```

Código Anterior

```
private static final Pattern acronym = Pattern.compile("[A-Z]+");

static List<String> converterCamelCase(String camelCaseString) {
    List<String> words = new ArrayList<>();

    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if (shouldBreak(camelCaseString, i)) {

            words.add(formatWord(camelCaseString.substring(0, i)));
            camelCaseString = camelCaseString.substring(i);
            i = 0;
        }
    }
```

```java
    words.add(formatWord(camelCaseString));
    System.out.println(String.join(", ", words));
    return words;
}

private static boolean shouldBreak(String camelCaseString, int i) {
    return isCapitalLetterOrNumber(camelCaseString.toCharArray()[i])
            && i != 0
            && (!isCapitalLetterOrNumber(camelCaseString.toCharArray()[i - 1])
            || (i < camelCaseString.length() - 1
            && !isCapitalLetterOrNumber(camelCaseString.toCharArray()[i + 1])));
}

private static boolean isCapitalLetterOrNumber(char c) {
    return (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9');
}

private static String formatWord(String word) {
    if (!acronym.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    return word;
}
```

Código Novo

```java
private static final Pattern acronym = Pattern.compile("[A-Z]+");
private static final Pattern startsWithNumber = Pattern.compile("^[0-9].*");

static List<String> converterCamelCase(String camelCaseString) {
    verifyValidString(camelCaseString);

    List<String> words = new ArrayList<>();

    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if (shouldBreak(camelCaseString, i)) {

            words.add(formatWord(camelCaseString.substring(0, i)));
            camelCaseString = camelCaseString.substring(i);
            i = 0;
        }
    }

    words.add(formatWord(camelCaseString));
    System.out.println(String.join(", ", words));
    return words;
}

private static void verifyValidString(String camelCaseString) {
    if (startsWithNumber.matcher(camelCaseString).matches())
        throw new InvalidNumberStartException("não deve começar com números");
}

private static boolean shouldBreak(String camelCaseString, int i) {
    return isCapitalLetterOrNumber(camelCaseString.toCharArray()[i])
            && i != 0
            && (!isCapitalLetterOrNumber(camelCaseString.toCharArray()[i - 1])
            || (i < camelCaseString.length() - 1
            && !isCapitalLetterOrNumber(camelCaseString.toCharArray()[i + 1])));
}

private static boolean isCapitalLetterOrNumber(char c) {
    return (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9');
}

private static String formatWord(String word) {
    if (!acronym.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    return word;
}
```

Descriçao

Adicionado verificaçao de caso de erro.

Ciclo: 7

Teste Adicionado

```
@Test(expected = InvalidSpecialCharactersException.class)
public void testInvalidSpecialCharacters() {
    String camelCaseString = "nome#Composto";
    CamelCase.converterCamelCase(camelCaseString);
}
```

Código Anterior

```
private static final Pattern acronym = Pattern.compile("[A-Z]+");
private static final Pattern startsWithNumber = Pattern.compile("^[0-9].*");

static List<String> converterCamelCase(String camelCaseString) {
    verifyValidString(camelCaseString);

    List<String> words = new ArrayList<>();

    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if (shouldBreak(camelCaseString, i)) {

            words.add(formatWord(camelCaseString.substring(0, i)));
            camelCaseString = camelCaseString.substring(i);
            i = 0;
        }
    }

    words.add(formatWord(camelCaseString));
    System.out.println(String.join(", ", words));
    return words;
}
```

```java
private static void verifyValidString(String camelCaseString) {
    if (startsWithNumber.matcher(camelCaseString).matches())
        throw new InvalidNumberStartException("não deve começar com números");
}

private static boolean shouldBreak(String camelCaseString, int i) {
    return isCapitalLetterOrNumber(camelCaseString.toCharArray()[i])
            && i != 0
            && (!isCapitalLetterOrNumber(camelCaseString.toCharArray()[i - 1])
            || (i < camelCaseString.length() - 1
            && !isCapitalLetterOrNumber(camelCaseString.toCharArray()[i + 1])));
}

private static boolean isCapitalLetterOrNumber(char c) {
    return (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9');
}

private static String formatWord(String word) {
    if (!acronym.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    return word;
}
```

Código Novo

```java
private static final Pattern ACRONYM = Pattern.compile("[A-Z]+");
private static final Pattern STARTS_WITH_NUMBER = Pattern.compile("^[0-9].*");
private static final Pattern ALLOWED_CHARS = Pattern.compile("^[0-9A-Za-zz]+$");

static List<String> converterCamelCase(String camelCaseString) {
    verifyValidString(camelCaseString);

    List<String> words = new ArrayList<>();

    for (int i = 0; i < camelCaseString.toCharArray().length; i++) {
        if (shouldBreak(camelCaseString, i)) {

            words.add(formatWord(camelCaseString.substring(0, i)));
            camelCaseString = camelCaseString.substring(i);
            i = 0;
        }
    }

    words.add(formatWord(camelCaseString));
    System.out.println(String.join(", ", words));
    return words;
}

private static void verifyValidString(String camelCaseString) {
    if (STARTS_WITH_NUMBER.matcher(camelCaseString).matches()) {
        throw new InvalidNumberStartException("não deve começar com números");
    }

    if (!ALLOWED_CHARS.matcher(camelCaseString).matches()) {
        throw new InvalidSpecialCharactersException("caracteres especiais não são
permitidos, somente letras e números");
    }
}

private static boolean shouldBreak(String camelCaseString, int i) {
    return isCapitalLetterOrNumber(camelCaseString.toCharArray()[i])
            && i != 0
            && (!isCapitalLetterOrNumber(camelCaseString.toCharArray()[i - 1])
            || (i < camelCaseString.length() - 1
            && !isCapitalLetterOrNumber(camelCaseString.toCharArray()[i + 1])));
}

private static boolean isCapitalLetterOrNumber(char c) {
    return (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9');
}

private static String formatWord(String word) {
    if (!ACRONYM.matcher(word).matches()) {
        word = word.toLowerCase();
    }
    return word;
}
```

Descriçao

Adicionada verificação de caracteres permitidos.