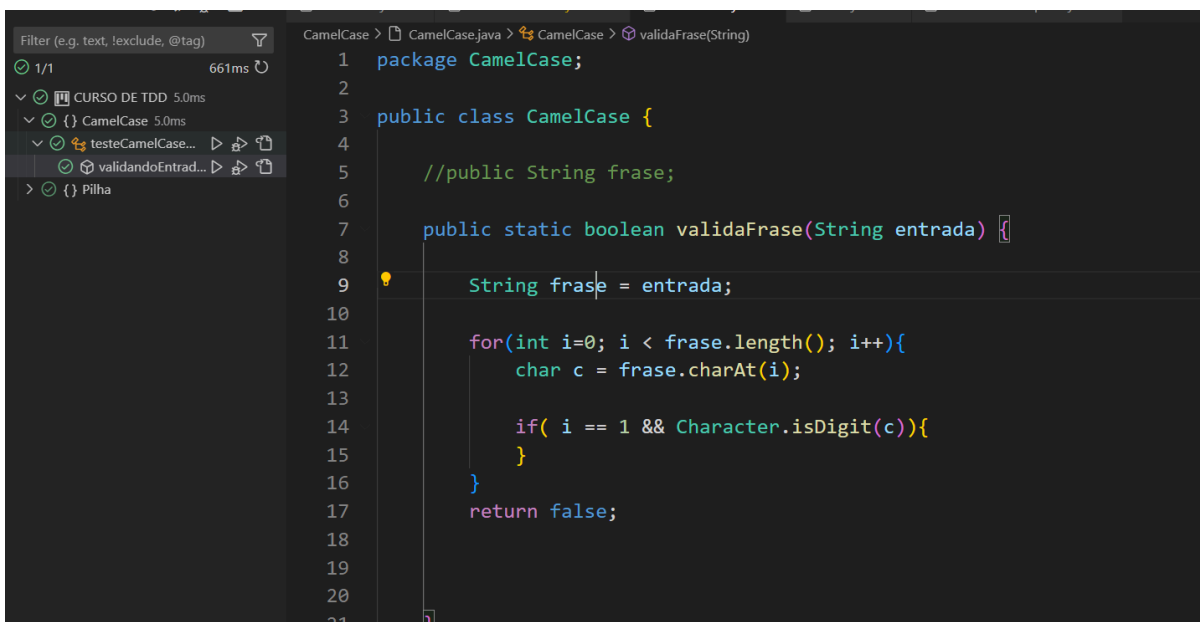


No primeiro momento comecei escrevendo um código que validasse a entrada, inicialmente chamada de 'frase'. Que não passou no teste, devido ao validaFrase não ter nenhuma implementação.



```
CamelCase > testeCamelCase.java > testeCamelCase
3 import static org.junit.Assert.assertFalse;
4 import static org.junit.Assert.assertTrue;
5
6 import org.junit.Test;
7
8 public class testeCamelCase {
9
10     @Test
11     public void validandoEntrada(){
12         assertFalse(CamelCase.validaFrase(string:"AndressaLo"));
13     }
14
15 }
16
```

Iniciei o processo de construção do código, tentando respeitar o teste e um dos critérios. E o teste inicial passou.



```
CamelCase > CamelCase.java > CamelCase > validaFrase(String)
1 package CamelCase;
2
3 public class CamelCase {
4
5     //public String frase;
6
7     public static boolean validaFrase(String entrada) {
8
9         String frase = entrada;
10
11         for(int i=0; i < frase.length(); i++){
12             char c = frase.charAt(i);
13
14             if( i == 1 && Character.isDigit(c)){
15             }
16         }
17         return false;
18
19
20
21 }
```

Tentei atender outros critérios com relação ao tipo de frase que não era aceito, começando com números e tendo carácter:

```

public static boolean validaFrase(String entrada) {
    String frase = entrada;
    for(int i=0; i < frase.length(); i++){
        char c = frase.charAt(i);
        if( i == 1 && Character.isDigit(c)){
            return false;
        }
        else if(!Character.isLetterOrDigit(c)){
            return false;
        }
    }

    return true;
}

```

Admitido que aqui acabei repetindo o processo vicioso de escrever o código antes do teste.

The screenshot shows an IDE with a test runner on the left and source code on the right. The test runner shows a list of tests: 'CURSO DE TDD' (15ms), 'CamelCase' (3.0ms), 'testeCamelCase' (3.0ms), 'validandoEntrada' (12ms), and 'Pilha' (12ms). The 'validandoEntrada' test is selected. The source code on the right is for 'CamelCase.java' and 'testeCamelCase.java'. The 'testeCamelCase.java' file contains the following code:

```

1 package CamelCase;
2
3 import static org.junit.Assert.assertFalse;
4 import static org.junit.Assert.assertTrue;
5
6 import org.junit.Test;
7
8 public class testeCamelCase {
9
10     @Test
11     public void validandoEntrada(){
12         assertTrue(CamelCase.validaFrase(entrada:"Andressa"));
13         assertFalse(CamelCase.validaFrase(entrada:"1111ansdd"));
14         assertFalse(CamelCase.validaFrase(entrada:"andr$ssaSil@"));
15     }
16
17 }
18

```

Iniciando a criação do teste para a função converterCamelCase:

```

@Test
public void validandoSaida(){
    ArrayList<String> listaEsperada = new ArrayList<>(Arrays.asList(...a:"Andressa", "Lorrayne"));
    ArrayList<String> listaAtual = CamelCase.converterCamelCase(original:"AndressaLorrayne");

    assertEquals(listaEsperada, listaAtual);
}

```

Código implementado verificando apenas as letras iniciais com UpperCase, ainda, não foi implementado para divisões que serão feitas com números, e ainda não estão saindo como minúsculas.

```
public static ArrayList<String> converterCamelCase(String original) {
    String frase = original;
    StringBuilder novaPalavra = new StringBuilder();
    ArrayList<String> camelCaseconvertido = new ArrayList<>();
    for(int i=0; i < frase.length(); i++){
        char c = frase.charAt(i);
        if(Character.isUpperCase(c)){
            if(novaPalavra.length() > 0 ){
                camelCaseconvertido.add(novaPalavra.toString());
                novaPalavra.setLength(newLength:0);
            }
            novaPalavra.append(c);
        }
    }
    if (novaPalavra.length() > 0) {
        camelCaseconvertido.add(novaPalavra.toString());
    }
    return camelCaseconvertido;
}
```

```
22
23 @Test
24 public void validandoSaida(){
25     ArrayList<String> listaEsperada = new ArrayList<>(Arrays.asList(...a:"andressa", "lorrayne"));
26     ArrayList<String> listaAtual = CamelCase.converterCamelCase(original:"AndressaLorrayne");
27
28     assertEquals(listaEsperada, listaAtual); Expected [[andressa, lorryne]] but was [[Andressa, lorryne]]
```

Expected [[andressa, lorryne]] but was [[Andressa, lorryne]] validandoSaida0

↑ ↓ ↺ | 🗑 🔄 ✕

Expected

-[andressa, lorryne]

Actual

+[Andressa, lorryne]

**Resultado da refatoração. Código:** As principais mudanças foram fazer a saída ser lower case, não havia no código anterior.

```
public static ArrayList<String> converterCamelCase(String original) {
    String frase = original;
    StringBuilder novaPalavra = new StringBuilder();
    ArrayList<String> camelCaseConvertido = new ArrayList<>();

    for(int i = 0; i < frase.length(); i++){
        char c = frase.charAt(i);
        //primeira condição pra formar nova palavra é ser UpperCase
        if(Character.isUpperCase(c)) {
            if (novaPalavra.length()>0) {
                camelCaseConvertido.add(novaPalavra.toString().toLowerCase());
                novaPalavra.setLength(0);
            }
        }
    }
}
```

```

        }
        novaPalavra.append(c);
    }
    if(novaPalavra.length()>0){
camelCaseConvertido.add(novaPalavra.toString().toLowerCase());

    }

    return camelCaseConvertido;

}

```

```

public static ArrayList<String> converterCamelCase(String original) {
    String frase = original;
    StringBuilder novaPalavra = new StringBuilder();
    ArrayList<String> camelCaseconvertido = new ArrayList<>();
    for(int i=0; i < frase.length(); i++){
        char c = frase.charAt(i);
        if(Character.isUpperCase(c)){
            if(novaPalavra.length() > 0 ){
                camelCaseconvertido.add(novaPalavra.toString().toLowerCase());
                novaPalavra.setLength(newLength:0);
            }
        }
        novaPalavra.append(c);
    }
    if (novaPalavra.length() > 0) {
        camelCaseconvertido.add(novaPalavra.toString().toLowerCase());
    }
    return camelCaseconvertido;
}

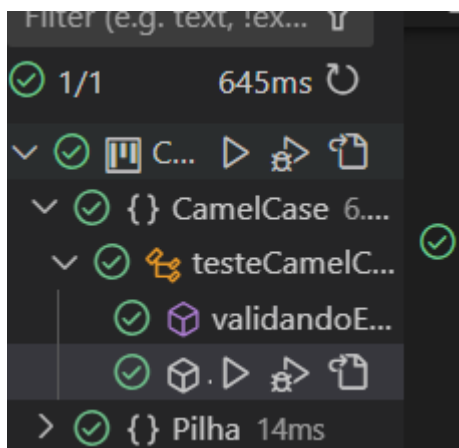
```

teste **passou**:

```

@Test
    public void validandoSaida(){
        ArrayList<String> listaEsperada = new
ArrayList<>(Arrays.asList("andressa","l","figueiredo"));
        ArrayList<String> listaAtual =
CamelCase.converterCamelCase("AndressaLFigueiredo");
        assertEquals(listaEsperada, listaAtual);
    }

```



Incluindo novo cenário no teste, aqui está incluso os demais critérios da tarefa, conseguir distinguir cada tipo de string: as começadas com maiúsculas, as que contem apenas maiusculas e as strings numéricas.

```
assertEquals(listaEsperada, listaAtual);

ArrayList<String> listaEsperada2 = new ArrayList<>(Arrays.asList(...a:"andressa","123", "cpf","lorrayne"));
ArrayList<String> listaAtual2 = CamelCase.converterCamelCase(original:"Andreessa123CPF Lorrayne");
assertEquals(listaEsperada2, listaAtual2);
}
```

Pequena mudança, que resultou no erro do teste:

```
29 char c = frase.charAt(i);
30 if(Character.isUpperCase(c) || Character.isDigit(c)){
31     if(novaPalavra.length() > 0 ){
32         camelCaseconvertido.add(novaPalavra.toString().toLowerCase());
33         novaPalavra.setLength(newLength:0);
34     }
35 }
36 novaPalavra.append(c);
37 }
38 if (novaPalavra.length() > 0) {
39     camelCaseconvertido.add(novaPalavra.toString().toLowerCase());
40 }
41 return camelCaseconvertido;
```

```
Esperada2 = new ArrayList<>(Arrays.asList(...a:"andressa","123", "cpf","lorrayne"));
Atual2 = CamelCase.converterCamelCase(original:"Andreessa123CPF Lorrayne");
assertEquals(listaEsperada2, listaAtual2); Expected [[andressa, 123, cpf, lorrayne]] but was [[andressa, 1, 2, 3, c, p, f, lorrayne]]
```

Refatoração do código:

```
public static ArrayList<String> converterCamelCase(String original) {
```

```

String frase = original;
StringBuilder novaPalavra = new StringBuilder();
ArrayList<String> camelCaseconvertido = new ArrayList<>();
for(int i=0; i < frase.length(); i++){
    char c = frase.charAt(i);
    char proxima = frase.charAt(i+1);
    if(Character.isUpperCase(c)){
        if(novaPalavra.length() > 0 ){
camelCaseconvertido.add(novaPalavra.toString().toLowerCase());
            novaPalavra.setLength(0);
        }
        if(Character.isUpperCase(proxima)){
            novaPalavra.append(c);
        }else{
            camelCaseconvertido.add(String.valueOf(c));
        }
    }else if (Character.isDigit(c)){
        if (novaPalavra.length() > 0) {
            camelCaseconvertido.add(novaPalavra.toString());
            novaPalavra.setLength(0);
        }else{
            novaPalavra.append(c);
        }
    }if (novaPalavra.length() > 0) {
camelCaseconvertido.add(novaPalavra.toString().toLowerCase());
    }
}

return camelCaseconvertido;
}

```

Erro no teste:

```
23  @Test
24  public void validandoSaida(){
25      ArrayList<String> listaEsperada = new ArrayList<>(Arrays.asList(...a:"addressa", "lorrayne"));
26      ArrayList<String> listaAtual = CamelCase.converterCamelCase(original:"AndressaLorrayne");    java.lang.String:

java.lang.StringIndexOutOfBoundsException: String index out of range: 16 at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:48) at java.base/java.lang.String.charAt(String.java:1513) at CamelCa... validand...
at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:48)
at java.base/java.lang.String.charAt(String.java:1513)
at CamelCase.CamelCase.converterCamelCase(CamelCase.java:30)
at CamelCase.testeCamelCase.validandoSaida(testeCamelCase.java:26)

27  assertEquals(listaEsperada, listaAtual);
```

## Refatorando:

```
public static ArrayList<String> converterCamelCase(String original)
{
    String frase = original;
    StringBuilder novaPalavra = new StringBuilder();
    ArrayList<String> camelCaseConvertido = new ArrayList<>();

    for(int i = 0; i < frase.length(); i++){
        char c = frase.charAt(i);
        //primeira condição pra formar nova palavra é ser
UpperCase
        if(Character.isUpperCase(c)){
            if (novaPalavra.length()>0) {
                camelCaseConvertido.add(novaPalavra.toString().toLowerCase());
                novaPalavra.setLength(0);
            }

            novaPalavra.append(c);
        }

        if(novaPalavra.length()>0){
            camelCaseConvertido.add(novaPalavra.toString().toLowerCase());
        }
    }
}
```

```
return camelCaseConvertido;
```

```
}
```

```
}
```

Teste de separação de palavras que começam com letra maiúscula e saída minúscula foi concluído com sucesso:

```
public void validandoSaida(){
    ArrayList<String> listaEsperada = new ArrayList<>(Arrays.asList(...a:"andressa","l", "s","figueiredo"));
    ArrayList<String> listaAtual = CamelCase.converterCamelCase(original:"AndressaLSFigueiredo");
    assertEquals(listaEsperada, listaAtual);
}
```

The screenshot shows an IDE with a project named 'CamelCase'. The file 'testeCamelCase.java' is open, showing a test method 'validandoSaida()' at line 23. The method contains two assertions comparing expected and actual lists of strings. The IDE's test runner shows that the test passed successfully. The bottom panel displays the test results, indicating that the test '1,validandoSaida(CamelCase.testeCamelCase)' passed at runtime.

```
Filter (e.g. text, !ex...)
0/0 204.8s
CURSO DE TDD...
CamelCase 7...
testeCamelC...
validandoE...
Pilha 14ms

21
22
23 @Test
24 public void validandoSaida(){
25     ArrayList<String> listaEsperada = new ArrayList<>(Arrays.asList(...a:"andressa","l", "s","figueiredo"));
26     ArrayList<String> listaAtual = CamelCase.converterCamelCase(original:"AndressaLSFigueiredo");
27     assertEquals(listaEsperada, listaAtual);
28
29     ArrayList<String> listaEsperada2 = new ArrayList<>(Arrays.asList(...a:"andressa","l", "s","figueiredo"));
30     ArrayList<String> listaAtual2 = CamelCase.converterCamelCase(original:"AndressaLSFigueiredo");
31     //assertEquals(listaEsperada2, listaAtual2);
32 }
33
34 }
35

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS PORTS
%TESTE 1,validandoSaida(CamelCase.testeCamelCase)
%RUNTIME17
```

**Criando novo cenário para testar String numéricas:**

```
ArrayList<String> listaEsperada2 = new
ArrayList<>(Arrays.asList("andressa","lorrayne","123", "cpfgg","lorrayne"));
    ArrayList<String> listaAtual2 =
CamelCase.converterCamelCase("AndressaLorrayne123CPFgGLorrayne");
    assertEquals(listaEsperada2, listaAtual2);
```



## Implementando código das Strings numéricas

```
public static ArrayList<String> converterCamelCase(String original) {
    String frase = original;
    StringBuilder novaPalavra = new StringBuilder();
    ArrayList<String> camelCaseConvertido = new ArrayList<>();

    for(int i = 0; i < frase.length(); i++){
        char c = frase.charAt(i);
        //primeira condição pra formar nova palavra é ser UpperCase
        if(Character.isUpperCase(c)){
            if (novaPalavra.length()>0) {

                camelCaseConvertido.add(novaPalavra.toString().toLowerCase());
                novaPalavra.setLength(0);
            }

            novaPalavra.append(c);

            if (Character.isDigit(c)) {
                if (novaPalavra.length() > 0) {

                    camelCaseConvertido.add(novaPalavra.toString().toLowerCase());
                    novaPalavra.setLength(0);
                }

                novaPalavra.append(c);

                while (i + 1 < frase.length() &&
Character.isDigit(frase.charAt(i + 1))) {
                    novaPalavra.append(frase.charAt(++i));
                }

                camelCaseConvertido.add(novaPalavra.toString());
                novaPalavra.setLength(0);
            }
        }
    }
}
```

```

    }

    if (novaPalavra.length() > 0) {

camelCaseConvertido.add(novaPalavra.toString().toLowerCase());

    }

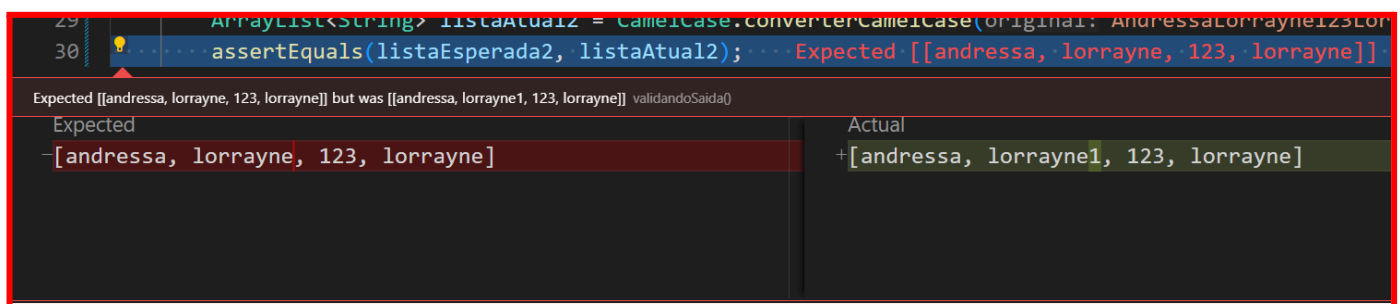
    return camelCaseConvertido;

}

}

```

Teste não passou.



Expected [[addressa, lorryne, 123, lorryne]] but was [[addressa, lorryne1, 123, lorryne]] validandoSaida()

Expected	Actual
-[addressa, lorryne, 123, lorryne]	+ [addressa1, lorryne, 123, lorryne]

Tentativas na refatoração, erro na lógica.

```

if (Character.isDigit(c)) {
    if (novaPalavra.length() > 0) {
        camelCaseConvertido.add(novaPalavra.toString());
        novaPalavra.setLength(0);
    }

    novaPalavra.append(c);

    while (i + 1 < frase.length() &&
Character.isDigit(frase.charAt(i + 1))) {
        novaPalavra.append(frase.charAt(++i));
    }

    camelCaseConvertido.add(novaPalavra.toString());
    novaPalavra.setLength(0);
}
}

```

Teste falhou da mesma forma:

Expected [[addressa, 123, lorryne, 123, lorryne]] but was [[addressa1, 123, lorryne1, 123, lorryne]] validandoSaida()	
Expected	Actual
-[addressa, 123, lorryne, 123, lorryne]	+[addressa1, 123, lorryne1, 123, lorryne]

Código refatorado para conseguir cumprir os três tipos de strings aceitos no array, palavras que foram iniciadas com maiúscula, que contém apenas maiúsculas, e string numérica. Foram feitas algumas alterações que antes não havia. Para corrigir o problema coloquei o `novaPalavra.append()` dentro do último `else` dentro do loop assim evitaria duplicações, e decidi transformar todas as strings em minúsculas apenas no final para tentar deixar o código mais limpo. Ainda não foi feita uma refatoração para clean code.

```
public static ArrayList<String> converterCamelCase(String original) {
    String frase = original;
    StringBuilder novaPalavra = new StringBuilder();
    ArrayList<String> camelCaseConvertido = new ArrayList<>();

    for(int i = 0; i < frase.length(); i++){
        char c = frase.charAt(i);
        //primeira condição pra formar nova palavra é ser UpperCase
        if(Character.isUpperCase(c)){
            if (novaPalavra.length()>0) {
                camelCaseConvertido.add(novaPalavra.toString());
                novaPalavra.setLength(0);
            }

        }

        //condição que inicia a string numerica
        if (Character.isDigit(c)) {
            if (novaPalavra.length() > 0) {
                camelCaseConvertido.add(novaPalavra.toString());
                novaPalavra.setLength(0);
            }

        }
        novaPalavra.append(c);
    }
}
```

```
        while (i + 1 < frase.length() &&
Character.isDigit(frase.charAt(i + 1))) {
            novaPalavra.append(frase.charAt(++i));
        }

        }else{

            novaPalavra.append(c);
        }

    }
    if(novaPalavra.length()>0){
        camelCaseConvertido.add(novaPalavra.toString());
    }

    camelCaseConvertido = camelCaseConvertido.stream()
        .map(String::toLowerCase)
        .collect(Collectors.toCollection(ArrayList::new));

    return camelCaseConvertido;

}

}
```

Teste passou!!!

The image shows a screenshot of an IDE with two tabs: 'testeCamelCase.java' and 'CamelCase.java'. The 'testeCamelCase.java' tab is active, displaying a Java class with a method 'validandoSaida()' and a test method '@Test public void validandoSaida()'. The code includes assertions for 'CamelCase.validaFrase()' and 'CamelCase.converterCamelCase()'. The IDE interface includes a sidebar with a project tree and a bottom toolbar.

```
16
17     assertTrue(CamelCase.validaFrase(entrada:"Andressa"));
18     assertFalse(CamelCase.validaFrase(entrada:"1111ansdd"));
19     assertFalse(CamelCase.validaFrase(entrada:"andr$ssaSil@"));
20 }
21
22 @Test
23 public void validandoSaida(){
24     ArrayList<String> listaEsperada = new ArrayList<>(Arrays.asList(...a:"andressa","l", "s","figueire
25     ArrayList<String> listaAtual = CamelCase.converterCamelCase(original:"AndressaSFigueiredo");
26     assertEquals(listaEsperada, listaAtual);
27
28     ArrayList<String> listaEsperada2 = new ArrayList<>(Arrays.asList(...a:"ana","g", "125","luz","588"
29     ArrayList<String> listaAtual2 = CamelCase.converterCamelCase(original:"AnaG125Luz588");
30     assertEquals(listaEsperada2, listaAtual2);
31 }
32
33
34
35
```