Coursera

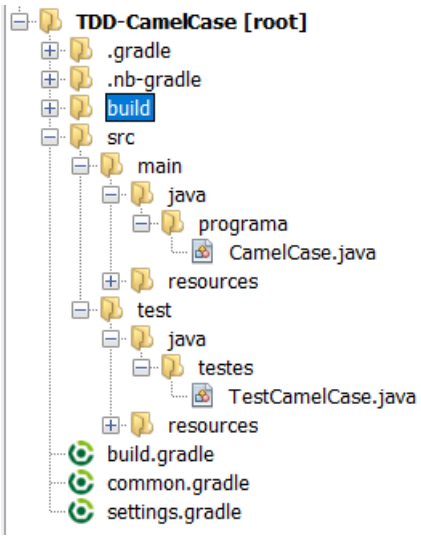
TDD – Desenvolvimento de Software Guiado por Testes por Instituto Tecnológico da Aeronáutica (ITA)

Atividades do Curso

Leonardo Simões
10/3/2019

Semana 1 – Atividade 1

Inicialmente foi criado um projeto Gradle com a Linguagem Java 8, chamado “TDD-CamelCase”. O Gradle foi utilizado porque na versão 8 do Java é necessário um gerenciador de dependências para adicionar e usar o JUnit 4.4. A IDE utilizada foi NetBeans IDE 8.3. As classes criadas foram TestCamelCase e CamelCase (após criar o primeiro teste). A classe CamelCase possui um método “public static List<String> converterCamelCase(String original)” conforme especificado no enunciado.



O processo de desenvolvimento do projeto foi criar testes unitários, que falharam, e então o código foi refatorado a fim de que passem nos testes.

Teste 1 – String sem palavras ou nula

Foi criada a classe TestCamelCase para testes. O primeiro teste criado foi para verificar os casos em que o converterCamelCase deve retornar uma lista vazia.

```
1   package testes;
2
3   //@author Leonardo Simões
4
5   import org.junit.*;
6   import static org.junit.Assert.*;
7
8   public class TestCamelCase {
9
10      @Test
11      public void semPalavras() {
12          assertTrue(CamelCase.converterCamelCase("").isEmpty());
13          assertTrue(CamelCase.converterCamelCase(null).isEmpty());
14      }
15
16  }
17
```

Refatoração 1

Foi criada a classe CamelCase e seu método converterCamelCase.

```
1   package programa;
2
3   //@author Leonardo Simões
4
5   import java.util.ArrayList;
6   import java.util.List;
7
8   public class CamelCase {
9
10      public static List<String> converterCamelCase(String original) {
11          List<String> p = new ArrayList<>();
12          return p;
13      }
14  }
15
```

Teste 2 – String com uma palavra

O segundo teste criado foi para verificar os casos em que o converterCamelCase deve retornar uma única palavra.

```
1   package testes;
2
3   //@author Leonardo Simões
4
5   import org.junit.*;
6   import static org.junit.Assert.*;
7   import programa.CamelCase;
8   import java.util.Arrays;
9
10  public class TestCamelCase {
11
12      @Test
13      public void semPalavras() {
14          assertTrue(CamelCase.converterCamelCase("").isEmpty());
15          assertTrue(CamelCase.converterCamelCase(null).isEmpty());
16      }
17
18      @Test
19      public void umaPalavra() {
20          assertEquals(CamelCase.converterCamelCase("nome"), Arrays.asList("nome"));
21          assertEquals(CamelCase.converterCamelCase("Nome"), Arrays.asList("nome"));
22          assertEquals(CamelCase.converterCamelCase("CPF"), Arrays.asList("CPF"));
23      }
24
25  }
```

Refatoração 2

Adicionado método para localizar os índices dos caracteres maiúsculos da string. O método converterCamelCase foi alterado.

```java
package programa;

//@author Leonardo Simões
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

    public static List<String> converterCamelCase(String original) {
        List<String> p = new ArrayList<>();
        if (original != null && !original.isEmpty()) {
            if (localizarMaiusculos(original).size() == 0) {
                p.add(original);
            } else if (localizarMaiusculos(original).size() == 1) {
                p.add(original.replace(original.charAt(0),
                        Character.toLowerCase(original.charAt(0))));
            } else {
                p.add(original);
            }
        }
        return p;
    }

    public static List<Integer> localizarMaiusculos(String original) {
        List<Integer> indices = new ArrayList<>();
        for (int i = 0; i < original.length(); ++i) {
            if (original.charAt(i) == Character.toUpperCase(original.charAt(i))) {
                indices.add(i);
            }
        }
        return indices;
    }

}
```

Teste 3 – String duas ou mais palavras

O segundo teste criado foi para verificar os casos em que o converterCamelCase deve retornar duas ou mais palavras.

```
1   package testes;
2
3   //@author Leonardo Simões
4   import org.junit.*;
5   import static org.junit.Assert.*;
6   import programa.CamelCase;
7   import java.util.Arrays;
8
9   public class TestCamelCase {
10
11      @Test
12      public void semPalavras() {
13          assertTrue(CamelCase.converterCamelCase("").isEmpty());
14          assertTrue(CamelCase.converterCamelCase(null).isEmpty());
15      }
16
17      @Test
18      public void umaPalavra() {
19          assertEquals(Arrays.asList("nome"), CamelCase.converterCamelCase("nome"));
20          assertEquals(Arrays.asList("nome"), CamelCase.converterCamelCase("Nome"));
21          assertEquals(Arrays.asList("CPF"), CamelCase.converterCamelCase("CPF"));
22      }
23
24      @Test
25      public void nPalavras() {
26          Arrays.asList("nome", "composto");
27          assertEquals(Arrays.asList("nome", "composto"),
28                  CamelCase.converterCamelCase("nomeComposto"));
29          assertEquals(Arrays.asList("nome", "composto"),
30                  CamelCase.converterCamelCase("NomeComposto"));
31          assertEquals(Arrays.asList("nome", "CPF"),
32                  CamelCase.converterCamelCase("NomeCPF"));
33          assertEquals(Arrays.asList("nome", "CPF", "contribuinte"),
34                  CamelCase.converterCamelCase("nomeCPFContribuinte"));
35          assertEquals(Arrays.asList("recupera", "10", "primeiros"),
36                  CamelCase.converterCamelCase("recupera10Primeiros"));
37      }
38
39  }
```

```
24      @Test
25      public void nPalavras() {
26          Arrays.asList("nome", "composto");
27          assertEquals(Arrays.asList("nome", "composto"),
28                  CamelCase.converterCamelCase("nomeComposto"));
29          assertEquals(Arrays.asList("nome", "composto"),
30                  CamelCase.converterCamelCase("NomeComposto"));
31          assertEquals(Arrays.asList("nome", "CPF"),
32                  CamelCase.converterCamelCase("numeroCPF"));
33          assertEquals(Arrays.asList("nome", "CPF", "contribuinte"),
34                  CamelCase.converterCamelCase("numeroCPFContribuinte"));
35          assertEquals(Arrays.asList("recupera", "10", "primeiros"),
36                  CamelCase.converterCamelCase("recupera10Primeiros"));
37      }
```

Refatoração 3

O método localizarMaiusculos foi refatorado (nome e implementação) para localizarPalavras.

Foi adicionado um método formatarPalavra para processar cada palavra separadamente.

Foi adicionado um método incluirIndice que contém a expressão booleana utilizada no método converterCamelCase.

O método converterCamelCase foi alterado.

```java
package programa;

//@author Leonardo Simões
import java.util.ArrayList;
import java.util.List;

public class CamelCase {

    public static List<String> converterCamelCase(String original) {
        List<String> p = new ArrayList<>();
        if (original != null && !original.isEmpty()) {
            List<Integer> indices = localizarPalavras(original);
            for (int i = 0; i < indices.size() - 1; ++i) {
                p.add(formatarPalavra(original.substring(indices.get(i), indices.get(i + 1))));
            }
        }
        return p;
    }

    public static List<Integer> localizarPalavras(String original) {
        List<Integer> indices = new ArrayList<>();
        indices.add(0);
        for (int i = 1; i < original.length() - 1; ++i) {
            if (incluirIndice(original, i))
                indices.add(i);
        }
        indices.add(original.length());
        return indices;
    }

    public static String formatarPalavra(String palavra) {
        if (palavra.equals(palavra.toUpperCase())) {
            return palavra;
        }
        return palavra.toLowerCase();
    }

    public static boolean incluirIndice(String original, int i) {
        return (!Character.isDigit(original.charAt(i)) && (original.charAt(i) == Character.toUpperCase(original.charAt(i))
                && (original.charAt(i + 1) == Character.toLowerCase(original.charAt(i + 1))
                || original.charAt(i - 1) == Character.toLowerCase(original.charAt(i - 1)))))
                || (Character.isDigit(original.charAt(i)) && !Character.isDigit(original.charAt(i - 1)));
    }
}
```

Resultados dos testes