ProjetoCamelCase

Breno Marques

Após criar o projeto foi desenvolvido vários ciclos TDD:

TDD Ciclo 01 - nome

- Implementação do teste:

```
public class CamelCaseTest {
	List<String> listaValoresAtuais;

	@Before
	public void iniciarListaValoresAtuais(){
		this.listaValoresAtuais = new ArrayList<>();
	}

	@Test
	public void umaPalavraMinuscula() {
		this.listaValoresAtuais = CamelCase.converterCamelCase("nome");
		assertEquals("nome", this.listaValoresAtuais.get(0));
	}
}
```

- Implementação na classe CamelCase para gerar a falha no teste:

```
public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		return null;
	}
}
```

- Implementação na classe CamelCase para corrigir o teste:

```
public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> palavras = new ArrayList<>();
		String umaPalavra = original;
		palavras.add(umaPalavra);
		return palavras;
	}
}
```

TDD Ciclo 02 - Nome

- Implementação do teste:

```
@Test
public void umaPalavraMaiusculo() {
	this.listaValoresAtuais = CamelCase.converterCamelCase("Nome");
	assertEquals(“nome", this.listaValoresAtuais.get(0));
}
```

- O novo teste foi executado com falha, e executado com sucesso após as alterações abaixo:

```
public class CamelCase {
	public static List<String> converterCamelCase(String original) {
		ArrayList<String> palavras = new ArrayList<>();
		String umaPalavra = original;
		palavras.add(formatarPalavra(umaPalavra));
		return palavras;
	}

	private static String formatarPalavra(String umaPalavra){
		return umaPalavra.toLowerCase();
	}
}
```

TDD Ciclo 03 - CPF

- Implementação do teste:

```
@Test
```

```
public void umaPalavraTodaMaiusculo() {
      this.listaValoresAtuais = CamelCase.converterCamelCase("CPF");
      assertEquals("CPF", this.listaValoresAtuais.get(0));
}
```

- O novo teste foi executado com falha, e executado com sucesso após as alterações abaixo:

```
public class CamelCase {
      public static List<String> converterCamelCase(String original) {
            ArrayList<String> palavras = new ArrayList<>();
            String umaPalavra = original;
            palavras.add(formatarPalavra(umaPalavra));
            return palavras;
      }

      private static String formatarPalavra(String umaPalavra) {
            String palavraTodaMaiuscula = umaPalavra.toUpperCase();
            if (umaPalavra.equals(palavraTodaMaiuscula))
                  return palavraTodaMaiuscula;
            else
                  return umaPalavra.toLowerCase();
      }
}
```

TDD Ciclo 04 - numeroCPF

- Implementação do caso de teste:

```
@Test
public void duasPalavrasUmaMinusculaOutraMaiusculo() {
      this.listaValoresAtuais = CamelCase.converterCamelCase("numeroCPF");
      assertEquals(2,listaValoresAtuais.size());
      assertEquals("numero", this.listaValoresAtuais.get(0));
      assertEquals("CPF", this.listaValoresAtuais.get(1));
}
```

- O novo teste foi executado com falha, e executado com sucesso após as alterações abaixo:

```
public class CamelCase {
      public static List<String> converterCamelCase(String original) {
            ArrayList<String> palavras = new ArrayList<>();
            char[] caracteres = original.toCharArray();
            int idxInicioProximaPalavra = 0;
            while (idxInicioProximaPalavra < original.length()) {
                  String umaPalavra = recuperarUmaPalavra(caracteres, idxInicioProximaPalavra);
                  palavras.add(formatarPalavra(umaPalavra));
                  idxInicioProximaPalavra += umaPalavra.length();
            }
            return palavras;
      }

      private static String formatarPalavra(String umaPalavra) {
            String palavraTodaMaiuscula = umaPalavra.toUpperCase();
            if (umaPalavra.equals(palavraTodaMaiuscula))
                  return palavraTodaMaiuscula;
            else
                  return umaPalavra.toLowerCase();
      }

      private static String recuperarUmaPalavra(char[] caracteres, int idxInicioProximaPalavra) {
            String umaPalavra = String.valueOf(caracteres[idxInicioProximaPalavra]);
            for (int idxCaracter = idxInicioProximaPalavra + 1; idxCaracter < caracteres.length; idxCaracter++) {
                  umaPalavra = umaPalavra.concat(String.valueOf(caracteres[idxCaracter]));
                  if ( (idxCaracter +1 < caracteres.length) && Character.isUpperCase(caracteres[idxCaracter]) != Character.isUpperCase(caracteres[idxCaracter + 1]))
                        break;
            }
            return umaPalavra;
      }
}
```

TDD Ciclo 04 - numeroCPFContribuinte

- Implementação do teste:

```
@Test
public void tresPalavras() {
      this.listaValoresAtuais = CamelCase.converterCamelCase("numeroCPFContribuinte");
      assertEquals(3,listaValoresAtuais.size());
```

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

- O novo teste foi executado com falha:“org.junit.ComparisonFailure: expected:<CPF[]> but was:<CPF[C]>”, e executado com sucesso após as alterações abaixo:

```
public class CamelCase {
    ...

    private static String recuperarUmaPalavra(char[] caracteres, int idxInicioProximaPalavra) {
        String umaPalavra = String.valueOf(caracteres[idxInicioProximaPalavra]);
        for (int idxCaracter = idxInicioProximaPalavra + 1; idxCaracter < caracteres.length; idxCaracter++) {
            umaPalavra = umaPalavra.concat(String.valueOf(caracteres[idxCaracter]));
            if (finalDaPalavra(caracteres, idxCaracter))
                break;
        }
        return umaPalavra;
    }

    private static boolean finalDaPalavra(char[] caracteres, int idxCaracterAtual) {
        int idxProximoCaracter = idxCaracterAtual + 1, idxTerceiroCaracter = idxProximoCaracter + 1;
        if (idxTerceiroCaracter < caracteres.length) {
            if (Character.isUpperCase(caracteres[idxCaracterAtual]) && Character.isLowerCase(caracteres[idxTerceiroCaracter]))
                return true;
            else if (Character.isUpperCase(caracteres[idxCaracterAtual]) != Character.isUpperCase(caracteres[idxProximoCaracter]))
                return true;
        }
        return false;
    }
}
```

TDD Ciclo 05 - recupera10Primeiros

- Implementação do teste:

```
@Test
public void tresPalavrasIncluindoNumeros() {
    this.listaValoresAtuais = CamelCase.converterCamelCase("recupera10Primeiros");
    assertEquals(3,listaValoresAtuais.size());
    assertEquals("recupera", this.listaValoresAtuais.get(0));
    assertEquals("10", this.listaValoresAtuais.get(1));
    assertEquals("primeiros", this.listaValoresAtuais.get(2));
}
```

- O novo teste foi executado com falha:“java.lang.AssertionError: expected:<3> but was:<2>” e executado com sucesso após as alterações a abaixo:

```
public class CamelCase {
    ...

    private static boolean finalDaPalavra(char[] caracteres, int idxCaracterAtual) {
        int idxProximoCaracter = idxCaracterAtual + 1, idxTerceiroCaracter = idxProximoCaracter + 1;
        if (idxTerceiroCaracter < caracteres.length) {
            if (Character.isLetter(caracteres[idxCaracterAtual]) != Character.isLetter(caracteres[idxProximoCaracter])) //Exemplo: recupera10Primeiros
                return true;
            if (Character.isUpperCase(caracteres[idxCaracterAtual]) && Character.isLowerCase(caracteres[idxTerceiroCaracter])) //Exemplo: numeroCPFContribuinte
                return true;
            else if (Character.isUpperCase(caracteres[idxCaracterAtual]) != Character.isUpperCase(caracteres[idxProximoCaracter])) //Exemplo: numeroCPF
                return true;
        }
        return false;
    }
}
```

TDD Ciclo 06 - 10Primeiros

- Implementação do teste:

```
@Test(expected=CamelCaseNaoComeçarComNumeroException.class)
public void nãoComecarComNumero() throws Exception {
    CamelCase.converterCamelCase("10Primeiros");
}
```

- O novo teste foi executado com falha:“java.lang.AssertionError: Expected exception: main.br.com.brolam.CamelCaseNaoComeçarComNumeroException”, e executado com sucesso após as alterações abaixo:

```
public class CamelCase {
	...

	public static List<String> converterCamelCase(String original) throws CamelCaseNaoComeçarComNumeroException {
		verificarPreRequisitos(original);
		ArrayList<String> palavras = new ArrayList<>();
		char[] caracteres = original.toCharArray();
		int idxInicioProximaPalavra = 0;
		while (idxInicioProximaPalavra < original.length()) {
			String umaPalavra = recuperarUmaPalavra(caracteres, idxInicioProximaPalavra);
			palavras.add(formatarPalavra(umaPalavra));
			idxInicioProximaPalavra += umaPalavra.length();
		}
		return palavras;
	}

	private static void verificarPreRequisitos(String original) throws CamelCaseNaoComecarComNumeroException {
		char primeiroCaracter = original.charAt(0);
		if (!Character.isLetter(primeiroCaracter))
			throw new CamelCaseNaoComeçarComNumeroException();
	}
}
```

TDD Ciclo 07 - nome#Composto

- Implementação do teste:

```
@Test(expected=CamelCaseCaracterInvalidoException.class)
public void caracterInvalido() throws Exception {
	CamelCase.converterCamelCase("nome#Composto");
}
```

- O novo teste foi executado com falha:“java.lang.AssertionError: Expected exception: main.br.com.brolam.CamelCaseCaracterInvalidoException”, e executado com sucesso após as alterações abaixo:

```
public class CamelCase {
		...
	private static void verificarPreRequisitos(String original)
			throws CamelCaseNaoComeçarComNumeroException, CamelCaseCaracterInvalidoException {
		char primeiroCaracter = original.charAt(0);
		if (!Character.isLetter(primeiroCaracter))
			throw new CamelCaseNaoComeçarComNumeroException();
		for (char caracter : original.toCharArray()) {
			if (!Character.isLetterOrDigit(caracter))
				throw new CamelCaseCaracterInvalidoException();
		}
	}
}
```