Relatório

→ Sequência um
Primeiramente criei a classe de teste e coloquei a função e o resultado da lista esperado.

```
package com.cabomaldade.testes;
import com.cabomaldade.app.SeparadoraCamelCase;
import org.junit.jupiter.api.Test;
import java.util.ArrayList;
import java.util.List;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class CamelCaseTestes {
	@Test
	public void converterDeCamelCaseParaLista() {
		SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
		List<String> expectativa = new ArrayList<>();
		expectativa.add("nome");
		assertEquals(expectativa, classeSeparadora.converterCamelCase("Nome"));
	}
}
```

Após, utilizando as declarações na classe teste, prossegui com a criação da classe a ser testada e seu método necessário para a operação, o método era o mais simples e só retornava o array vazio, fazendo com que o teste não passasse, mas compilasse.

```
package com.cabomaldade.app;
import java.util.ArrayList;
import java.util.List;
public class SeparadoraCamelCase {
	public List<String> converterCamelCase(String original) {
		List<String> listaSeparada = new ArrayList<String>();
		return listaSeparada;
	}
}
```

Então fiz o teste passar, tornando uma string que começou com uma letra maíuscula em uma string minúscula.

```
package com.cabomaldade.app;
import java.util.ArrayList;
import java.util.List;
public class SeparadoraCamelCase {
	public List<String> converterCamelCase(String original) {
		List<String> listaSeparada = new ArrayList<>();
		listaSeparada.add(original.toLowerCase());
		return listaSeparada;
	}
}
```

→ Sequência 2

Criado o teste para tentar obter mais de um item na lista, levando em consideração a segunda letra maíuscula.

```
@Test
public void converterDeCamelCaseParaListaAcimaDeUma() {
	SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
	List<String> expectativa = new ArrayList<>();
	expectativa.add("nome");
	expectativa.add("composto");
	assertEquals(expectativa, classeSeparadora.converterCamelCase("NomeComposto"));
}
```

Após isso, foi preciso mexer na classe para que ela conseguisse fazer o split, tornando assim uma string em uma lista de palavras.

```
package com.cabomaldade.app;
import java.util.ArrayList;
import java.util.List;
public class SeparadoraCamelCase {
	public List<String> converterCamelCase(String original) {
		List<String> listaSeparada = new ArrayList<>();
		String temp = String.valueOf(original.charAt(0));
		for (int i=1; i< original.length(); i++) {
			if (Character.isLowerCase(original.charAt(i))) {
				temp += original.charAt(i);
			} else {
				listaSeparada.add(temp.toLowerCase());
				temp = "";
				temp += original.charAt(i);
			}
		}
		listaSeparada.add(temp.toLowerCase());
		return listaSeparada;
	}
}
```

Então precisei refatorar para poder atender a questão de 10 linhas

```
package com.cabomaldade.app;
import java.util.ArrayList;
import java.util.List;
public class SeparadoraCamelCase {
	private List<String> listaSeparada = new ArrayList<>();
	public List<String> converterCamelCase(String original) {
		if (!validaCamelCase(original)){
			listaSeparada.add(original.toLowerCase());
		} else {
			adicionaNaLista(original, 0);
		}
		return this.listaSeparada;
	}
	private void adicionaNaLista(String original, int inicio){
		for (int i=1; i< original.length(); i++) {
			if (Character.isUpperCase(original.charAt(i))) {
				this.listaSeparada.add(original.substring(inicio,i).toLowerCase());
				inicio = original.substring(inicio,i).length();
```

```
                } else if (i == original.length() -1){
                        this.listaSeparada.add(original.substring(inicio,i+1).toLowerCase());
                }
        }
    }
    private boolean validaCamelCase(String original) {
        for (int i=1; i< original.length(); i++) {
            if (Character.isUpperCase(original.charAt(i))) {
                return true;
            }
        }
        return false;
    }
}
```

Com a refatoração, os testes continuarão passando.

→ Sequência 3

```
@Test
public void converterDeCamelCaseParaListaContendoNumeros() {
    SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
    List<String> expectativa = new ArrayList<>();
    expectativa.add("recupera");
    expectativa.add("10");
    expectativa.add("primeiros");
    assertEquals(expectativa, classeSeparadora.converterCamelCase("recupera10Primeiros"));
}
```

Teste com números falhou, partindo para implementar na classe.

```
package com.cabomaldade.app;
import java.util.ArrayList;
import java.util.List;
public class SeparadoraCamelCase {
    private List<String> listaSeparada = new ArrayList<>();
    public List<String> converterCamelCase(String original) {
        if (!validaCamelCase(original) ){
            listaSeparada.add(original.toLowerCase());
        } else if (!original.matches(".*\\d.*")){
            adicionaPalavras(original.split("(?=[A-Z])"));
        } else {
            adicionaLetrasENumeros(original.split("(?=[A-Z])"));
        }
        return this.listaSeparada;
    }
    private void adicionaPalavras(String[] palavrasEmCamelCase){
        int i = 0;
        while (i < palavrasEmCamelCase.length){
            listaSeparada.add(palavrasEmCamelCase[i].toLowerCase());
            i ++;
        }
    }
    private void adicionaLetrasENumeros(String[] letrasComNumeros){
        int i = 0;
        while (i < letrasComNumeros.length){
            if (letrasComNumeros[i].matches(".*\\d.*")) {
```

```
                    trataArrayLetrasENumeros(letrasComNumeros[i].split("(?<=\\d)(?=\\D)|(?
<=\\D)(?=\\d)"));
                } else{
                    listaSeparada.add(letrasComNumeros[i].toLowerCase());
                }
                i ++;
            }
        }
        private void trataArrayLetrasENumeros(String[] array){
            int i = 0;
            while (i < array.length){
                listaSeparada.add(array[i].toLowerCase());
                i ++;
            }
        }
        private boolean validaCamelCase(String original) {
            for (int i=1; i< original.length(); i++) {
                if (Character.isUpperCase(original.charAt(i)) ||
Character.isDigit(original.charAt(i))) {
                    return true;
                }
            }
            return false;
        }
}
```

Agora foi possível passar os testes, esse é o código com três testes passando.
A difilculdade maior é colocar as funções dentro de 10 linhas, então uso a parte do Character para validar e também a parte de String com expressões regulares para resumir a busca.

→ Seção 4 – implementação do teste onde validará o que é inválido de entrada de string

Os testes implementados para checar se é válido o imput não passaram.

```
@Test
public void checarSeStringEValidaComecaComNumero() {
    SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
    List<String> expectativa = new ArrayList<>();
    expectativa.add("inválido");
    assertEquals(expectativa, classeSeparadora.converterCamelCase("10Primeiros"));
}
@Test
public void checarSeStringEValidaTemCaracterEspecial() {
    SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
    List<String> expectativa = new ArrayList<>();
    expectativa.add("inválido");
    assertEquals(expectativa, classeSeparadora.converterCamelCase("nome#Composto"));
}
```

Adicionado então código na classe para fazer o teste passar

```
package com.cabomaldade.app;
import java.util.ArrayList;
import java.util.List;
public class SeparadoraCamelCase {
    private final List<String> listaSeparada = new ArrayList<>();
    public List<String> converterCamelCase(String original) {
```

```
            if(validaStringDeEntrada(original)){
                if (!validaCamelCase(original) ){
                    listaSeparada.add(original.toLowerCase());
                } else if (!original.matches(".*\\d.*")){
                    adicionaPalavras(original.split("(?=[A-Z])"));
                } else {
                    adicionaLetrasENumeros(original.split("(?=[A-Z])"));
                }
            } else {
                this.listaSeparada.add("inválido");
            }
        return this.listaSeparada;
    }
    private void adicionaPalavras(String[] palavrasEmCamelCase){
        int i = 0;
        while (i < palavrasEmCamelCase.length){
            listaSeparada.add(palavrasEmCamelCase[i].toLowerCase());
            i ++;
        }
    }
    private void adicionaLetrasENumeros(String[] letrasComNumeros){
        int i = 0;
        while (i < letrasComNumeros.length){
            if (letrasComNumeros[i].matches(".*\\d.*")) {
                trataArrayLetrasENumeros(letrasComNumeros[i].split("(?<=\\d)(?=\\D)|(?
<=\\D)(?=\\d)"));
            } else{
                listaSeparada.add(letrasComNumeros[i].toLowerCase());
            }
            i ++;
        }
    }
    private void trataArrayLetrasENumeros(String[] array){
        int i = 0;
        while (i < array.length){
            listaSeparada.add(array[i].toLowerCase());
            i ++;
        }
    }
    private boolean validaCamelCase(String original) {
        for (int i=1; i< original.length(); i++) {
            if (Character.isUpperCase(original.charAt(i)) ||
Character.isDigit(original.charAt(i))) {
                return true;
            }
        }
        return false;
    }
    private boolean validaStringDeEntrada(String stringParaValidacao) {
        if (Character.isDigit(stringParaValidacao.charAt(0)) ||
validaCaracteresEspeciais(stringParaValidacao)) {
            return false;
        } else {
            return true;
        }
    }
    private boolean validaCaracteresEspeciais(String stringParaValidacao) {
        int i = 0;
        while (i < stringParaValidacao.length()){
```

```
                if (!Character.isDigit(stringParaValidacao.charAt(i)) && !
Character.isLetter(stringParaValidacao.charAt(i))) {
                        return true;
                }
                i ++;
        }
        return false;
    }
```

→ Com isso chego ao final dos testes e agora fica o tempo para refatoração, tanto do código de teste como para a classe.

Abaixo segue o código antes da refatoração com os testes toods passando

→ Classe de teste

```
package com.cabomaldade.testes;
import com.cabomaldade.app.SeparadoraCamelCase;
import org.junit.jupiter.api.Test;
import java.util.ArrayList;
import java.util.List;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class CamelCaseTestes {
    @Test
    public void converterDeCamelCaseParaLista() {
        SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
        List<String> expectativa = new ArrayList<>();
        expectativa.add("nome");
        assertEquals(expectativa, classeSeparadora.converterCamelCase("Nome"));
    }
    @Test
    public void converterDeCamelCaseParaListaAcimaDeUma() {
        SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
        List<String> expectativa = new ArrayList<>();
        expectativa.add("nome");
        expectativa.add("composto");
        assertEquals(expectativa, classeSeparadora.converterCamelCase("NomeComposto"));
    }
    @Test
    public void converterDeCamelCaseParaListaContendoNumeros() {
        SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
        List<String> expectativa = new ArrayList<>();
        expectativa.add("recupera");
        expectativa.add("10");
        expectativa.add("primeiros");
        assertEquals(expectativa,
classeSeparadora.converterCamelCase("recupera10Primeiros"));
    }
    @Test
    public void checarSeStringEValidaComecaComNumero() {
        SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
        List<String> expectativa = new ArrayList<>();
        expectativa.add("inválido");
        assertEquals(expectativa, classeSeparadora.converterCamelCase("10Primeiros"));
    }
```

```
	@Test
	public void checarSeStringEValidaTemCaracterEspecial() {
		SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
		List<String> expectativa = new ArrayList<>();
		expectativa.add("inválido");
		assertEquals(expectativa,
classeSeparadora.converterCamelCase("nome#Composto"));
	}
}
```

Após refatoração da classe teste, temos o código abaixo.

```
package com.cabomaldade.testes;
import com.cabomaldade.app.SeparadoraCamelCase;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.Test;
import java.util.ArrayList;
import java.util.List;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class CamelCaseTestes {
	SeparadoraCamelCase classeSeparadora = new SeparadoraCamelCase();
	List<String> expectativa = new ArrayList<>();
	@Test
	public void converterDeCamelCaseParaLista() {
		expectativa.add("nome");
		assertEquals(expectativa, classeSeparadora.converterCamelCase("Nome"));
	}
	@Test
	public void converterDeCamelCaseParaListaAcimaDeUma() {
		expectativa.add("nome");
		expectativa.add("composto");
		assertEquals(expectativa, classeSeparadora.converterCamelCase("NomeComposto"));
	}
	@Test
	public void converterDeCamelCaseParaListaContendoNumeros() {
		expectativa.add("recupera");
		expectativa.add("10");
		expectativa.add("primeiros");
		assertEquals(expectativa,
classeSeparadora.converterCamelCase("recupera10Primeiros"));
	}
	@Test
	public void checarSeStringEValidaComecaComNumero() {
		expectativa.add("inválido");
		assertEquals(expectativa, classeSeparadora.converterCamelCase("10Primeiros"));
	}
	@Test
	public void checarSeStringEValidaTemCaracterEspecial() {
		expectativa.add("inválido");
		assertEquals(expectativa,
classeSeparadora.converterCamelCase("nome#Composto"));
	}
	@AfterEach
	void aposCada() {
		expectativa.clear();
	}
}
```

Partindo para a Classe SeparadoraCamelCase.java – o código antes da refatoração é:

```
package com.cabomaldade.app;
import java.util.ArrayList;
import java.util.List;
public class SeparadoraCamelCase {
	private final List<String> listaSeparada = new ArrayList<>();
	public List<String> converterCamelCase(String original) {
		if(validaStringDeEntrada(original)){
			if (!validaCamelCase(original) ){
				listaSeparada.add(original.toLowerCase());
			} else if (!original.matches(".*\\d.*")){
				adicionaPalavras(original.split("(?=[A-Z])"));
			} else {
				adicionaLetrasENumeros(original.split("(?=[A-Z])"));
			}
		} else {
			this.listaSeparada.add("inválido");
		}
		return this.listaSeparada;
	}
	private void adicionaPalavras(String[] palavrasEmCamelCase){
		int i = 0;
		while (i < palavrasEmCamelCase.length){
			listaSeparada.add(palavrasEmCamelCase[i].toLowerCase());
			i ++;
		}
	}
	private void adicionaLetrasENumeros(String[] letrasComNumeros){
		int i = 0;
		while (i < letrasComNumeros.length){
			if (letrasComNumeros[i].matches(".*\\d.*")) {
				trataArrayLetrasENumeros(letrasComNumeros[i].split("(?<=\\d)(?=\\D)|(?
<=\\D)(?=\\d)"));
			} else{
				listaSeparada.add(letrasComNumeros[i].toLowerCase());
			}
			i ++;
		}
	}
	private void trataArrayLetrasENumeros(String[] array){
		int i = 0;
		while (i < array.length){
			listaSeparada.add(array[i].toLowerCase());
			i ++;
		}
	}
	private boolean validaCamelCase(String original) {
		for (int i=1; i< original.length(); i++) {
			if (Character.isUpperCase(original.charAt(i)) ||
Character.isDigit(original.charAt(i))) {
				return true;
			}
		}
		return false;
	}
	private boolean validaStringDeEntrada(String stringParaValidacao) {
		if (Character.isDigit(stringParaValidacao.charAt(0)) ||
validaCaracteresEspeciais(stringParaValidacao)) {
```

```
                return false;
            } else {
                return true;
            }
        }
        private boolean validaCaracteresEspeciais(String stringParaValidacao) {
            int i = 0;
            while (i < stringParaValidacao.length()){
                if (!Character.isDigit(stringParaValidacao.charAt(i)) && !
Character.isLetter(stringParaValidacao.charAt(i))) {
                    return true;
                }
                i ++;
            }
            return false;
        }
}
```

E o código após refatoração ficou assim:

```
package com.cabomaldade.app;
import java.util.ArrayList;
import java.util.List;
public class SeparadoraCamelCase {
        private final List<String> listaSeparada = new ArrayList<>();
        public List<String> converterCamelCase(String original) {
                if(validaStringDeEntrada(original)){
                    preencherLista(original);
                } else {
                    this.listaSeparada.add("inválido");
                }
            return this.listaSeparada;
        }
        private void preencherLista(String original){
            if (!validaCamelCase(original) ){
                listaSeparada.add(original.toLowerCase());
            } else if (!original.matches(".*\\d.*")){
                adicionaPalavras(original.split("(?=[A-Z])"));
            } else {
                adicionaLetrasENumeros(original.split("(?=[A-Z])"));
            }
        }
        private void adicionaPalavras(String[] palavrasEmCamelCase){
            int i = 0;
            while (i < palavrasEmCamelCase.length){
                listaSeparada.add(palavrasEmCamelCase[i].toLowerCase());
                i ++;
            }
        }
        private void adicionaLetrasENumeros(String[] letrasComNumeros){
            int i = 0;
            while (i < letrasComNumeros.length){
                if (letrasComNumeros[i].matches(".*\\d.*")) {
                    trataArrayLetrasENumeros(letrasComNumeros[i].split("(?<=\\d)(?=\\D)|(?
<=\\D)(?=\\d)"));
                } else{
                    listaSeparada.add(letrasComNumeros[i].toLowerCase());
                }
                i ++;
```

```
            }
      }
      private void trataArrayLetrasENumeros(String[] array){
            int i = 0;
            while (i < array.length){
                  listaSeparada.add(array[i].toLowerCase());
                  i ++;
            }
      }
      private boolean validaCamelCase(String original) {
            for (int i=1; i< original.length(); i++) {
                  if (Character.isUpperCase(original.charAt(i)) ||
Character.isDigit(original.charAt(i))) {
                        return true;
                  }
            }
            return false;
      }
      private boolean validaStringDeEntrada(String stringParaValidacao) {
            if (Character.isDigit(stringParaValidacao.charAt(0)) ||
validaCaracteresEspeciais(stringParaValidacao)) {
                  return false;
            } else {
                  return true;
            }
      }
      private boolean validaCaracteresEspeciais(String stringParaValidacao) {
            int i = 0;
            while (i < stringParaValidacao.length()){
                  if (!Character.isDigit(stringParaValidacao.charAt(i)) && !
Character.isLetter(stringParaValidacao.charAt(i))) {
                        return true;
                  }
                  i ++;
            }
            return false;
      }
}
```

Não foi preciso alterar tanto, já que já havia refatorado da outra vez, mas obedeceu as 10 linhas por função.

Termino aqui este trabalho, achei bem interessante à partir do segundo e terceiro teste implementado, é bom ver que por várias vezes o teste quebrou os demais na hora de tentar fazer passar, isso me fez desenvolver pensando no código já implementado também, foi uma ótima experiência e satisfatória para um desafio que tive, bem grande por sinal.