1- Teste adicionado

```
	@Test
	public void camelCase() {
		List<String> lista = Arrays.asList("roberto", "Higor");
		assertEquals(lista, Camel.converterCamelCase("robertoHigor"));
	}	}
```

Codigo antes:

```
	private static List<String> camel;

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

	public static List<String> retornar() {
		return camel;
}
```

Código depois:

```
public class Camel {

	private static List<String> camelc = new ArrayList<>();

	public static List<String> converterCamelCase(String original) {
		String[] guardar = original.split("(?<!^)(?=[A-Z])");
		for (int i = 0; i < guardar.length; i++) {
			camelc.add(guardar[i]);
		}
		return camelc;
	}
}
```

Descrição do que foi feito:

Foi criado a classe do teste e o incio do código.

2- Teste adicionado

```
	@Before
	public void before(){
		Camel.limpar();
	}
e

	@Test(expected = CamelNumeroException.class)
	public void camelNaoNumero() {
		List<String> lista = Arrays.asList("1roberto", "Higor");
		assertEquals(lista,
Camel.converterCamelCase("1robertoHigor"));
```

```
    }
```

Código antes:

```
    private static List<String> camelc = new ArrayList<>();

    public static List<String> converterCamelCase(String original) {
        String[] guardar = original.split("(?<!^)(?=[A-Z])");
        for (int i = 0; i < guardar.length; i++) {
            camelc.add(guardar[i]);
        }
        return camelc;
    }
}
```

Código depois:

```
public class Camel {

    private static List<String> camelc = new ArrayList<>();

    public static List<String> converterCamelCase(String original) {

        if (Character.isDigit(original.charAt(0))) {
            throw new CamelNumeroException("Não pode começar com
números");
        } else {

            String[] guardar = original.split("(?<!^)(?=[A-Z])");
            for (int i = 0; i < guardar.length; i++) {
                camelc.add(guardar[i]);
            }
        }
        return camelc;
    }

    public static void limpar() {
        camelc.clear();
    }
}
```

Descrição do que foi feito:
Foi criado um novo teste para verificar se contém um numero antes da palavra. Foi preciso criar o método limpar pois como a lista é estática, ela estava guardando os 4 valores. O Teste dava certo caso o código atual estivesse rodando mas caso fosse removido a exception, ele dava erro.

3 – Refatoração
Código antes:

```
}public class Camel {

    private static List<String> camelc = new ArrayList<>();
```

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

		if (Character.isDigit(original.charAt(0))) {
			throw new CamelNumeroException("Não pode começar com
números");
		} else {

			String[] guardar = original.split("(?<!^)(?=[A-Z])");
			for (int i = 0; i < guardar.length; i++) {
				camelc.add(guardar[i]);
			}
		}
		return camelc;
	}

	public static void limpar() {
		camelc.clear();
	}
}
```

Código depois:

```
	public static List<String> converterCamelCase(String original) {
		validar(original);
		String[] guardar = original.split("(?<!^)(?=[A-Z])");
		for (int i = 0; i < guardar.length; i++) {
			camelc.add(guardar[i]);
		}
		return camelc;
	}

	public static void limpar() {
		camelc.clear();
	}

	public static String validar(String original) {
		if (Character.isDigit(original.charAt(0))) {
			throw new CamelNumeroException("Não pode começar com
números");
		} else {
			return original;
		}
	}
}
```

Descrição da alteração:

Foi criado o método validar para diminuir o tamanho do converterCamelCase e simplificar o funcionamento dele.