TDD – Testing Driven Development - Curso TDD – ITA/Coursera

Semana 01 – Atividade final / Aluno: Vagner Panarello Filho

1. Criação do corpo da classe “SplitStrings” com o método estático “converterCamelCase”. A classe de testes do Junit também foi criada neste momento iniciando com os seguinte métodos.

    a. “t_output_itensCount()”: Esse método tem como objetivo testar se o método retornava uma lista com o mesmo número de ítens relacionado as divisões CamelCase.
    b. “t_Input_nomeComposto()”: Esse método testava as strings de cada ítem da lista. Se eram condizentes com a string de entrada. Foi uma das únicas que funcionou inicialmente porque até então não havia implementado a rotina na qual coloca todas as palavras para caixa baixa.
    c. Outros métodos de teste foram implementados, mas nenhum passava no teste nesse ponto pois a Classe “SplitStrings” ainda estava muito incompleta.

2. Primeira estratégia era implementar primeiramente o algorítimo no qual separava uma string unica em strings menores e divididas pelo seus camel cases.

    a. Primeiro método adotado foi de utilizar o método “String.toCharArray()” no qual um Array contendo os char da string eram avaliados um a um em um loop for. Desiti de utilizar esse metodo porque estava ficando muito confuso e não estava separando as palavras corretamente.
    b. A estratégia que foi adotada e se manteve até o final do projeto foi, criar uma Array de nome “upperCaseIndexes” na qual armazena somente os índices somente das letras maiúsculas da String de entrada. Com esses valores elas são separadas através do método “original.substring(start,end)” e armazenados na lista de retorno.

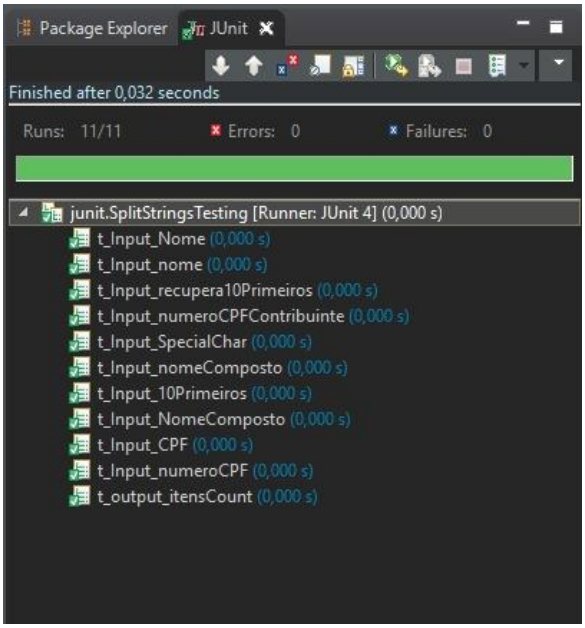
3. Após mais algumas linhas de cógido foram adicionadas com objetivo de redimensionar a array “upperCaseIndexes” com o tamanho exato da quantidade de indices necessários.

```
int[] upperCaseIndexes = new int[arrayIndex];

for(int i = 0; i < arrayIndex; i++){                // resize array
    upperCaseIndexes[i] = _upperCaseIndexes[i];
}
_upperCaseIndexes = null;                           // free first temp array
```

4. Com objetivo de passar nos últimos testes foi adicionado o último pedaço de código com objetivo de passar para lowerCase somente as palavras que com apenas um caracter em UpperCase. Desta maneira mantendo cadeias de letras em UpperCase em maiúsculo, como CPF, e passando para minúsculo o restante das palavras. Após isso todos os teste começaram a funcionar corretamente.

```
// put to lowCase when needed
List<String> retList = new ArrayList<>();

for (String word : _retList) {
    int upperCaseCharsNumber = 0;
    for(int i = 0; i < word.length(); i++) {
        int c = word.charAt(i);
        if (c >= 65 && c <= 90) upperCaseCharsNumber++; // count number of upperCase char in a word
    }
    if (upperCaseCharsNumber <= 1 ) retList.add(word.toLowerCase());
    else retList.add(word);
}
```

5. O desenvolvimento do projeto foi uma tarefa demaziada complicada. Quase que uma programação em baixo nível, por ser necessário intensa manipulação de Arrays puras e juntamente tomando-se todos os cuidados para que não ocorressem erros de “IndexOutOfBoundsException”. Foram criados 11 testes, todos passaram corretamente os final da implementação. Os últimos a serem criados e implementados foram os de testes de exceções para caracteres especiais e strings começadas com números. O último resultado dos testes segue abaixo:



Refatorando o Projeto

6. Como próxima etapa foi decidido optimizar o código, inicialmente os fragmentando em pequenas pedaços, que antes estavam em um único bloco, em métodos separados. São eles:

    a. **“private static void stringValidating(String camelCaseString)”**
    Responsável pela validação da String em CamelCase. Utilizando “IF” juntamente com uma verificação de intervalos numéricos baseado na tabela ASCII de caracteres. Caso de caracteres especiais ou numéricos em posições improprias geram exceções.
    b. **“private static String splitString(String camelCaseString)”**
    Principal modificação durante o processo de refatoração. Foi decidido substituir o antigo padrão de separação de string baseado em IF’s e intervalos numéricos da tabela ASCII, para o uso de expressões regulares. O que torou o código infinitamente mais simples.
    c. **“private static List<String> charCasesNormalizer(List<String> wordsList)”**
    Não foi modificado do que era antes da refatoração, responsável por passar para caixa baixa todas as palavras, exceto siglas.

7. Após a fatoração e fragmentação do projeto, todos os testes passaram exatamente como já estavam antes do processo de fatoração. Nenhuma modificação foi feita nos métodos de testes.