Trabalho Prático: *The SpellChecker*

Prof. Daniel Conrado

1 Introdução

Neste trabalho, você deve criar um projeto em Java para servir como corretora ortográfica (*spellchecker*) para alguma aplicação de texto (como um processador de texto, p. ex.).

1.1 Objetivo

O objetivo deste trabalho prático é trabalhar a manipulação de coleções, e fornecer condições para vocês treinarem programação orientada a objetos e o pensamento lógico.

1.2 Método

Você deve criar o projeto do zero, ele deve-se chamar "spelling". Você deve criar uma classe chamada **SpellChecker**. Eu vou te dar um arquivo para servir de dicionário, contendo uma boa quantidade de palavras em inglês. Também te darei o código-fonte de uma classe chamada **DictReader**. Essa classe pode ler o arquivo de dicionário e fornecer o dicionário como um objeto **ArrayList** de **Strings**.

Findo o projeto, você deve enviar para mim, via Classroom, o código-fonte das classes, ou seja, os arquivos com extensão **.java**, sem estarem compactados.

2 Exercícios

Exercício 1. A sua classe **SpellChecker** vai conseguir obter um dicionário em inglês de um objeto **DictReader**. A classe **SpellChecker** deve ter os seguintes métodos públicos:

`public int numberOfWords()`: deve retornar o número de palavras no dicionário.

`public boolean isKnownWord(String word)`: retorna verdadeiro se e somente se a palavra informada por parâmetro for encontrada no dicionário.

`public boolean allKnown(ArrayList<String> words)`: este método retorna verdadeiro se e somente se todas as palavras contidas na lista passada por parâmetro (words) também estão contidas no dicionário.

Exercício 2. Implemente mais dois métodos:

`public ArrayList<String> wordsStartingWith(String prefix)`: este método retorna uma lista de todas as palavras do dicionário que começam com o prefixo dado.

`public ArrayList<String> wordsContaining(String text)`: este método retorna uma lista de todas as palavras do dicionário que contém a String passada por parâmetro.

Certifique-se de que esses métodos sejam *case-insensitive*. Ou seja, se eu quiser procurar por palavras que começam com "gilb", ele deve considerar p. ex. a palavra "Gilbert".

Exercício 3. Implemente os métodos:

`public void insert(String newWord)`: este método insere a dada palavra no dicionário. Note que a palavra só poderá ser inserida se ela ainda não existir no dicionário. Se ela já existir, o método não faz nada. Atenção: certifique-se de que a ordem alfabética do dicionário continue válida.

`public boolean remove(String newWord)`: este método remove a palavra informada do dicionário. Se a palavra for removida com sucesso, retorne true. Caso contrário (p. ex. a palavra não existe no dicionário), retorne false.

`public void save()`: este método salva o dicionário no sistema de arquivos. Este método não é difícil, porque existe um método no objeto **DictReader** que você pode usar para isso.

Exercício 4. Implemente os métodos:

`public boolean isPalindrome(String word)`: retorna verdadeiro se e somente se a palavra informada é uma palíndrome. Note que a palavra só vai ser considerada palíndrome se ela também existir no dicionário.

`public ArrayList<String> anagrams(String word)`: retorne uma lista com todas as palavras que são anagramas da palavra informada. Os anagramas são tratados como *case-insensitive*.

`public ArrayList<String> difference(ArrayList<String> dictionary)`: dado um outro dicionário como parâmetro, compare-o com o seu, e retorna uma lista de todas as palavras que estão em um dos dicionários e que não estão no outro.

Exercício 5. Implemente o método:

`public int distance(String word1, String word2)`: compute a distância entre as duas palavras informadas. Essa distância é comumente usada em dicionários para sugerir correções ortográficas. Quando as aplicações de texto sugerem correções para uma palavra, elas costumam mostrar as palavras que têm as menores distâncias dela.

O cômputo deve usar a **Distância de Levenshtein**. Você vai encontrar mais detalhes nos links:

`http://www.cut-the-knot.org/do_you_know/Strings.shtml`

`http://www-igm.univ-mlv.fr/~lecroq/seqcomp/node2.html`