ASSIGNMENT 5
Comparing DNA Strings

COMP-202, Summer 2010

Due: Tuesday, June 15th, 2010

1 Introduction

The field of **bioinformatics** uses computers and algorithms to solve problems from biology. This is useful, because in biology scientists often need to analyze large amounts of data. One example is the analysis of DNA using computers. We can use them to compare DNA strings, and find how similar they are, which gives us a measure of how closely related two species are. Evolution is based on mutations of DNA strings. An atomic mutation is either an insertion, deletion or substitution of one of the base pairs that makes up the DNA string, denoted by c, g, a and t. We can look at two strings and find out what the least number of mutations are to go from one string to the other. This notion in general is called the **edit distance** between two strings. It measures how many atomic edits (insertions, deletions, substitutions) one needs to go from the one string to the other:

	foo	bar	oar
foo	0	3	3
bar	3	0	1
oar	3	1	0

There exists an **algorithm** to compute the edit distance, and you will need to implement it in this assignment. The algorithm is presented below in something called **pseudo code**. Pseudo code is a precise description of an algorithm using a syntax that is similar to a programming language, but it doesn't really resemble any particular one. It might use mathematical notation, slightly different syntax, like ← or `:=` as assignment operators, or just indentation but no braces, or subscripts to denote array accesses. In `Java`, indexing always starts at `0`, but in pseudo code the first index may be `1`. Sometimes English is used to describe certain aspects of an algorithm. There are no fixed conventions on how to write pseudo code, as long as it is precise and makes clear what is meant, without being actual code.

Algorithm: **Edit Distance**
input: two character strings $s_{1..m}$, $t_{1..n}$
output: the edit distance between s and t
uses: an integer array $d_{0..m,0..n}$

for $i = 0..m$
  $d_{i,0} \leftarrow i$
for $j = 0..n$
  $d_{0,j} \leftarrow j$

for $i = 1..m$
  for $j = 1..n$
    if $s_i = t_j$ then
      $d_{i,j} \leftarrow d_{i-1,j-1}$
    else
      $d_{i,j} \leftarrow 1 + \min(d_{i-1,j}, d_{i,j-1}, d_{i-1,j-1})$

return $d_{m,n}$

In this assignment we will read DNA sequences from files, and compare them using the edit distance algorithm. The sequences describe the same gene from different species, and comparing them will give us an idea how closely they are related. Once we found the relationships, we can draw a genealogical tree.

2 Specification

This assignment will consist of three classes, one just providing static methods, one a blue print for objects, and one just providing a main method. You have to follow the interfaces (method names, input/output types) exactly as specified, but you may add your own private helper methods. As usual, you should test every single method you add. You may do so by adding a main method to every class that you write, which tetsts the functionality of the class.

2.1 StringUtils

We will need a little helper library providing us with some useful String methods. The edit distance will be computed here. All methods are static. `getDistance` takes two strings, and returns their edit distance as an integer. `filter` takes two strings, a text and a mask. It will return a new string which is the same as the text, but only retaining characters that are present in the mask (`filter("hi how are you?","hoAi")` → `"hihoo"`).

2.2 DNAString

Our first class desribing on object, describes a dna sequence. Internally it stores the dna sequence and the species name as a String, as private members. Our interface (the public members) consists only of instance methods. Implement the methods in the following table:

method name	**parameters**	**function and return**
<constructor>	Two strings: a species and a dna sequence	constructs the dna string using the arguments
<constructor>	A string denoting a file name	reads the dna sequence from the given filename
`toString`	-	returns a String representation of the dna String
`getSpecies`	-	returns the species as a String
`getMutations`	Another `DNAString`	returns the edit distance to the other dna string

Hint: if you look at the sequence files you will see how they are formatted. When reading them from file, you have to make sure that you only store the species description, and the dna sequence - but the sequence should only consist of the letters c, g, a and t.

Note that the `toString` method allows us to directly print a `DNAString` using `System.out.println`. If you implement the method, you can directly print the object, without converting it to a `String`. This is because `println` will call the `toString` method automatically, if the argument is an object. Try to see what gets printed if you do not add the `toString` method.

2.3 Main

The `Main` class only provides a `main` method. It should read two file names as command line arguments (from the `args` parameter), load the dna sequences that exist at that location and compare them. The program should print out the comparison in a user-friendly way. If less than two commmand line arguments are specified, the method should print a welcome and a description of how to use the program.

2.4 Analyze data

Use your program to compare the n dna Sequence files that can be found on the webpage. Make a table to show them. Try to come up with a genealogical tree. Keep this short and simple.

Submit both your program, and your analysis of the data.