XML and modern techniques of content management 2010/11

Task 3

The story is based od an idea given by Mr. Pawła Bednarza (but spoiled by me ☺).

The story

Implement a computer program aimed to help at learning of foreign languages. The input of the program are two XML data files:

1. Dictionary which contains set of words: entries from one language and glosses, describing their meanings (possible in various languages). Please refer to the scheme **schemat_A_2010_Zad3XML.xsd** for more details.
   Please note that you cannot assume that the file is small (can fit in memory) or that it is sorted. You can assume, however, that it does not contain duplicate entries.

2. The total results of previous tests for selected words. The file contains two kinds of information: how many times a word has been tested (TestNumber) and how many answers were correct (SuccessNumber). It can be assumed that the file is small and that word entries are placed in the same order as in the dictionary (in particular, that the corresponding entries in the dictionary exist.). The structure of the file is defined by scheme **schemat_B_2010_Zad3XML.xsd**. Please note that the languages are set for the whole file.

The program should generate a file with questions, which is consistent with schema given in file **schemat_C_2010_Zad3.xsd** (please refer to the schema for more details). The file is to contain two sets of questions:

1. The list of multiple choice questions each asking on meaning of one word, there are four possible answers (one correct).

2. The list of word descriptions.

The number of tested words is given by a program parameter, the language of questions is given by languageOfDescription attribute in .results file.

Program call parameters

Another program call parameters mean:

1. The input file containing a dictionary.
2. The input file containing results of previous tests.
3. The output file containing generated tests.
4. A number that specifies the number of words being tested in the generated test (maximum, see algorithm)
5. Number that specifies the location from which to begin search in the dictionary (<= 1, see algorithm)

Class containing the main method is located in the default package, and bear the name of TestGenerator.

Algorithm ☺

0. The first words to be tested are selected from a file with the results of previous tests, in the order of occurrence in the file.
1. If in the file with the results of previous tests there are not enough words, the rest are taken from the file containing dictionary, starting from the place indicated by the "number that specifies the location from which to begin searching in the dictionary" parameter. Previously selected words are skipped (so are the words which do not have a description In the language of descripiton), dictionary should be treated as a cyclic list. If there is not enough words in the dictionary – all the words from the dictionary, which have a description in the language, are being put in the test.
2. The number of variants of answers is 4. Incorrect answers are retrieved from other word descriptions (or generated in another smart way).
3. In the second part of the test (the list of word descriptions), the previously listed words are arranged from the best to the least known.
4. In assessing whether a word is known look at
    a. The percentage of correct answers

        ("How many times given the correct answer" / "how many times the word has been tested")

    b. If the ratio is equal to looking at "how many times the word has been tested"

    c. If also the ratio is equal assume that the more familiar is the first word in the order of the dictionary.
5. For the words not listed in the file with the results:

"How many times given the correct answer" = "how many times the word was tested" = 0

Additional requirements

1. The program should be written in Java; It will be compiled and run in the environment Sun Java SE 6.
2. Dictionary file can not be read into memory (in particular, the program will be tested with a large paper input and limited memory), you should use SAX and STAX. DOM API should also be used (e.g. when processing the file with results).
3. The program might read the input files twice.
4. Method of processing files is up to the author's decision (but look at 2 above).
5. Program should handle namespaces, output files should be compatible with the schema . **schemat_C_2010_Zad3XML.xsd**
6. Source code files should use UTF-8.
7. Program should check whether the languages in dictionary and the result file are consistent (attributes anguageOfEntry, languageOfWord).

Uwagi końcowe:

1. The solution should be sent by 26 January 2011 (17:59) at the address kedar@mimuw.edu.pl, the title of the email should be"XML 2010, Z3, Solution", attached to the e-mail should be a ZIP- containing a directory named name_surname.
2. Program code should be documented (comprehensively, but without exaggeration), preferably in a way that allows automatic generation of documentation (JavaDoc)). In particular, at the beginning of each file should contain information about its author (name, surname).
3. A script should be provided for quick compilation and running of the program (eg, ant) – a README file must provide appropiate information on the program/script usage.
4. Each started 12 hours of delay means 1 point less.
5. Please check the FAQ.
6. In generated file it should be assumed that the schema is located in the same directory as the generated file.
7. It can be assumed that the input documents are valid.
8. Please check that the output document validate against the schema.