XML i nowoczesne technologie zarządzania treścią, rok akademicki 2010/11

Zadanie 3

Bajka

Bajka jest oparta na pomyśle zgłoszonym przez pana Pawła Bednarza (jednak bardzo zepsutym przeze mnie:-).

Należy zaimplementować program wspomagający naukę języków obcych. Program działa w oparciu o dwa pliki z danymi (zgodnymi ze schematem **schemat_A_2010_Zad3XML.xsd** oraz schematem **schemat_B_Zad3XML.xsd**):

1. Słownik zawierający: hasła (wszystkie z tego samego języka) oraz opisy (z określeniem kategorii, języka opisu). Uwaga: nie można zakładać , że plik jest mały (zmieści się w pamięci) lub że jest posortowany. Można zakładać, że nie zawiera duplikatów haseł.

2. Łączne wyniki użytkownika w konkurencji wprowadzanie hasła na podstawie podanego opisu oraz w konkurencji wybierania poprawnej definicji spośród czterech podanych. Plik zawiera dla każdego słowa dwie informacje: ile razy słowo było testowane, ile razy udzielono poprawnej odpowiedzi. Można założyć że plik jest mały i że wpisy są w nim umieszczone w takiej samej kolejności jak w słowniku (w szczególności że odpowiadające wpisy w słowniku istnieją). Wyniki są dla danego słownika i dla opisów w konkretnym języku.

Program ma generować plik z pytaniami, zgodny z **schemat_C_2010_Zad3.xsd**) w pliku tym na początku ma znajdować się w:

1. Lista haseł wraz z 4 opisami (w tym jeden prawidłowy) – test wyboru

2. Lista opisów wyżej wymienionych haseł (pytania utrwalające, słowa te same co powyżej) – pytania otwarte/test utrwalający

Liczba słów testowanych ma być określona przez parametr wywołania programu.

Kolejność jest określona przez podany poniżej algorytm.

Język opisów jest zadany przez parametr languageOfDescription w pliku z wynikami użytkownika.

Parametry wywołania programu

Kolejne parametry wywołania programu oznaczają:

0. Plik wejściowy zawierający dane słownikowe.
1. Plik wejściowy zawierający wyniki użytkownika w testach.
2. Plik wyjściowy zawierający wygenerowany test.
3. Liczba określająca liczbę testowanych wyrażeń w wygenerowanym teście (maksymalną, patrz algorytm)

4. Liczba określająca od którego miejsca w słowniku zaczynać poszukiwanie (<=1, patrz algorytm)

Klasa zawierająca metodę main ma znajdować się w domyślnym pakiecie i nosić nazwę TestGenerator.

Algorytm ☺

1. Pierwsze słowa do przetestowania są wybierane z pliku z wynikami w kolejności zadanej przez ten plik (test wyboru).

2. Jeżeli w pliku z wynikami nie ma wystarczającej liczby słów to kolejne słowa do testu wyboru są brane ze słownika począwszy od miejsca wyznaczonego przez „Liczba określająca od którego miejsca w słowniku zaczynać poszukiwanie". Gdy znajdziemy się już w tym miejscu słownika to pomijane są słowa już wybrane (z pliku z wynikami) oraz nie posiadające opisu w wymaganym języku. Słownik jest traktowany jak lista cykliczna, w przypadku gdy liczba słów w słowniku posiadających opisy w wymaganym języku jest mniejsza od parametru wywołania „Liczba określająca liczbę testowanych wyrażeń w wygenerowanym teście" – brane są wszystkie słowa ze słownika posiadających odpowiednie opisy.

3. Liczba wariantów odpowiedzi wynosi 4. Niepoprawne odpowiedzi mają być pobierane z innych haseł (lub generowane w inny sposób gdy innych haseł nie ma). Sposób oraz kolejność odpowiedzi dowolna.

4. W drugiej części testu (test utrwalający) słowa są umieszczone od najbardziej znanych do najmniej znanych.

5. Przy ocenie czy słowo jest znane patrzymy na

    a. Procent poprawnych odpowiedzi
    („ile razy udzielono poprawnej odpowiedzi"/ „ ile razy słowo było testowane")

    b. W przypadku gdy iloraz jest równy patrzymy na „ile razy słowo było testowane"

    c. W przypadku gdy powyższe nie rozstrzyga przyjmujemy że bardziej znane jest pierwsze słowo według kolejności słownika (miejsca w pliku).

    Oczywiście dla słów nie umieszczonych w pliku z wynikami przyjmujemy

    „ile razy udzielono poprawnej odpowiedzi"=„ ile razy słowo było testowane" =0

Uwagi i wymagania w stosunku do implementacji

1. Program powinien być napisany w Javie, kompilować się i działać w środowisku Sun Java SE 6 (tak będzie testowany).
2. Dokumenty ze słownikiem nie mogą być wczytywane w całości do pamięci (w szczególności program będzie testowany z dużym dokumentem wejściowym i ograniczoną pamięcią), ma zostać użyty **SAX lub StAX**. Interfejs **DOM** ma zostać również użyty (np. do czytania pliku z wynikami).
3. Dopuszczalne jest maksymalnie dwukrotne czytanie dokumentów wejściowych.

4. Sposób przetwarzania dokumentów zależy od autora rozwiązania (to znaczy, że np. można stosować kolekcje dostępne w Javie, zamiast operować na drzewie DOM).
5. Należy obsługiwać przestrzenie nazw, pliki wynikowe powinny być zgodne ze stanowiącym załącznik do niniejszego zadania schematem **schemat_C_2010_Zad3.xsd**.
6. Plik(i) źródłowe kodu programu mają używać kodowania UTF-8.
7. Sytuacje błędne np: niezgodność languageOfEntry w słowniku z „ languageOfWord" w pliku z wynikami powinny być wykryte – a odpowiedni komunikat powinien pojawić się na standardowym wyjściu programu. Nie trzeba sprawdzać tego o czym powiedziano, że można zakładać.

Uwagi końcowe:

1. Rozwiązanie powinno zostać przesłane do dnia 26 stycznia 2011 (17:59) z serwera students na adres kedar@mimuw.edu.pl, tytuł emaila powinien brzmieć „XML 2010, Z3, rozwiazanie", załącznik do emaila powinien stanowić plik-archiwum ZIP o nazwie ab123456.zip zawierające katalog o nazwie ab123456 (ab123456 należy zastąpić loginem na komputerze students).
2. Kod programu powinien być udokumentowany (wyczerpująco, ale oczywiście bez przesady komentarzy ma być znacznie mniej niż kodu– najlepiej w sposób umożliwiający automatyczną generację dokumentacji (JavaDoc)). W szczególności na początku każdego pliku powinna znajdować się informacja o jego autorze (imię, nazwisko, login na students).
3. Ma zostać dostarczony skrypt umożliwiający szybką kompilację i uruchomienie (np. ant) wraz z opisem uruchomienia (README).
4. Za przesłanie rozwiązania w terminie późniejszym naliczone zostaną punkty karne (każde rozpoczęte 12 godzin – 1 punkt).
5. Proszę sprawdzać FAQ, mogą się tam pojawić informacje np. o modyfikacji schemy (mam jednak nadzieję, że nie będzie to konieczne).
6. Lokalizacja schematu w dokumencie wyjściowym ma być ustalona na katalog bieżący.
7. Można założyć, że dokumenty wejściowe są zgodne ze schematem.
8. Dokumenty wynikowe mają się walidować !