

XML i nowoczesne technologie zarządzania treścią, rok akademicki 2008/09

Zadanie 3

Bajka

Firma informatyczna zatrudnia pracowników biorących udział w projektach. Co jakiś czas zbierane są, w celu przetworzenia, godzinki (dane mówiące o tym ile godzin dany pracownik przepracował w danym projekcie danego dnia). Firma chce otrzymać program który mając na wejściu dane o pracownikach, projektach, godzinkach, dostępną kwotę pieniędzy oraz datę wygeneruje dwa pliki:

- listę wypłat dla pracowników
- listę godzinek (z zapisanym statusem wskazującym na to czy godzinka została opłacona, a jeżeli nie to z jakiej przyczyny)

Sposób wyliczenia wypłat zostanie opisany w dalszej części zadania.

Parametry wywołania programu

Kolejne argumenty oznaczają ścieżki do plików lub inne napisy (kwotę pieniędzy, datę).

Wszystkie dokumenty XML mają być zgodne ze schematem **schemat2008Zad3.xsd**.

Kolejne parametry wywołania programu oznaczają:

0. plik zawierający dane pracowników firmy (element główny: pracownicy).
1. plik zawierający opisy projektów realizowanych przez firmę (element główny: projekty).
2. plik z godzinkami, godzinka zawiera informację o pracy pracownika nad projektem w danym dniu (element główny: godzinki).
3. liczbę naturalną, budżet na opłacenie pensji (np. 1000).
4. datę zapadalności w formacie rrrr-mm-dd (np. 2009-01-10).
5. plik wynikowy (wygenerowany przez program) z pracownikami firmy (element główny: pracownicyOut)
6. plik wynikowy (wygenerowany przez program) z godzinkami (element główny: godzinkiOut)

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

Algorytm ☺

Program ma opracować listę wypłat dla pracowników w miesiącu daty zapadalności (np. gdy data zapadalności to 2009-01-10, to miesiącem tym jest styczeń 2009) oraz przetworzyć każdą godzinkę (nadać jej status).

Budżet jest wydawany w następujący sposób:

1. W pierwszej kolejności opłacane są pensje pracowników (zawarte w elementach pensja).
2. W przypadku gdyby budżet nie wystarczył na opłacenie wszystkich pensji pracowników, program kończy działanie (bez generowania jakichkolwiek plików, wypisując na standardowe

wyjście komunikat: „Budżet wynosi <kwota>, a na opłacenie pensji pracowników potrzeba <kwota>. Zwiększ budżet”.

3. W drugiej kolejności opłacane są **godziny** z poprzednich miesięcy (poprzedzających miesiąc daty zapadalności, wszystkie takie godziny są traktowane jako nadgodziny).
4. W trzeciej kolejności opłacane są **nadgodziny** z bieżącego miesiąca (z miesiąca daty zapadalności).
5. W przypadku gdyby pieniędzy wystarczyło tylko na opłacenie części godzin z poprzednich miesięcy (lub nadgodzin z bieżącego miesiąca) to w pierwszej kolejności mają być opłacone wpisy umieszczone wcześniej w pliku z godzinkami (czyli przy określaniu priorytetu najpierw liczy się termin-godziny z poprzednich miesięcy, a potem miejsce w pliku).
6. Godziny z miesięcy następujących po bieżącym nie są opłacane (np. gdy podana data zapadalności to 2009-01-10, to godziny pracy z 2009-02-11 nie są opłacane, a godziny z 2009-01-31 zostaną opłacone, o ile budżet jest wystarczający).
7. W **bieżącym** miesiącu każdy pracownik przepracowuje bezpłatnie 160*wymiar etatu godzin (np. zatrudniony na pół etatu 80 godzin, zatrudniony na 2/3 etatu 107 godzin – czyli wynik mnożenia zaokrąglamy w górę).
8. Za jedną nadgodzinę z bieżącego miesiąca (godzinę z minionego miesiąca) pracownik otrzymuje kwotę określoną przez element stawka.
9. W przypadku gdy część godzin z wpisu godzinka (w pliku z godzinkami):
 - a. powinna być przepracowana przez pracownika bezpłatnie (w ramach etatu), a część płatnie w ramach nadgodzin.lub
 - b. może zostać opłacona, a część nie może (z powodu przekroczenia budżetu) to wpis w pliku wynikowym powinien być podzielony na dwie lub maksymalnie trzy części (rozliczoną w etacie, opłaconą, nieopłaconą, tzn. stany: wEtacie, opłacony, brakFunduszy).
10. W przypadku gdy pieniędzy wystarczyło na opłacenie wszystkich pensji pracowników (sumy liczb zawartych w elementach pensja), program:
 - a. wypisuje na standardowe wyjście następującą informację „Budżet wynosi <kwota> wydano <kwota>, do uregulowania należności za miesiąc bieżący i poprzednie potrzeba <kwota>.”
 - b. w wygenerowanym pliku wynikowym z pracownikami firmy powinny znaleźć się informacje o należnych wypłatach (w elemencie wypłata). Lista pracowników powinna być posortowana po numerach PESEL. Data wypłaty dla wszystkich pracowników powinna być taka sama i być równa ostatniemu dniu miesiąca daty zapadalności (np. gdy jako parametr przekazano „2009-01-10”, data ta powinna być równa „2009-01-31”).
 - c. w pliku wynikowym z godzinkami powinny znaleźć się informacje o opłaceniu pracy. Konkretnie dopuszczalna wartość atrybutu status to: wEtacie, opłacony, brakPracownikaLubProjektu, brakFunduszy, zobowiazanieWPrzyszlosci. Kolejność wpisów w dokumentach wynikowych powinna być następująca:
 - i. zgodna z kolejnością wpisów w dokumencie wejściowym.
 - ii. w przypadku podziału wpisu (na skutek różnego sposobu rozliczenia godzin) powstałe wpisy są uporządkowane w następującej kolejności (wg statusu): wEtacie, opłacony, brakFunduszy.

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 z pracownikami oraz z projektami mogą być wczytywane/zapisywane w standardowy sposób **za pomocą DOM lub JAXB**.
3. Dokumenty z godzinkami 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**.
4. Dopuszczalne jest dwukrotne czytanie dokumentów wejściowych.
5. Sposób przetwarzania dokumentów zależy od autora rozwiązania (to znaczy, że np. można stosować kolekcje dostępne w Javie, zamiast operować na drzewie DOM).
6. Należy obsługiwać przestrzeń nazw, pliki wynikowe powinny być zgodne ze stanowiącym załącznik do niniejszego zadania schematem.

Uwagi końcowe:

1. W uzasadnionym (np. niechęcią do Javy lub miłością do innego języka programowania) przypadku, można prosić o pozwolenie na implementację zadania w innym języku programowania niż Java (prośba powinna zawierać podstawowe informacje o planowanej do użycia technologii – np. linki do stron z opisem bibliotek). Prośbę tę należy kierować do autora zadania (kedar@mimuw.edu.pl).
2. Rozwiązanie powinno zostać przesłane do dnia **30 stycznia 2009** (23:59) z serwera students na adres kedar@mimuw.edu.pl, tytuł emaila powinien brzmieć „XML 2008, zad 3, rozwiązanie”, załącznik do emaila powinien stanowić plik-archiwum ZIP o nazwie ab123456.zip zawierające katalog o nazwie ab123456 (ab123456 należy zastąpić loginem na komputerze students).
3. 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).
4. Bardzo proszę by, w miarę możliwości, rozwiązanie było zawarte w pojedynczym pliku lub został dostarczony skrypt umożliwiający szybką kompilację (np. ant).
5. Za przesłanie rozwiązania w terminie późniejszym naliczone zostaną punkty karne (1 dzień – 1 punkt, 2 dni – 2 punkty), rozwiązania nadesłane później sprawdzane nie będą.
6. Proszę sprawdzać [FAQ](#), mogą się tam pojawić informacje np. o modyfikacji schemy (mam jednak nadzieję, że nie będzie to konieczne).
7. Proszę zapoznać się z plikami przykładowymi (pracownicy.xml, projekty.xml, godzinki.xml) oraz zmodyfikować je w taki sposób by przetestować wszystkie możliwe typy sytuacji np.:
 - a. brak pracownika
 - b. brak projektu
 - c. inna kolejność elementów, np. podelementów w elemencie projekt (typ all!)
 - d. konieczność podziału wpisu godzinka na dwie lub trzy części
8. Lokalizacja schematu w dokumentach wyjściowych może być (w zależności od decyzji autora rozwiązania)
 - a. taka sama jak w odpowiadającym dokumencie wejściowym
 - b. ustalona na plik **schemat2008Zad3.xsd** (w katalogu dokumentu, tzn. bez ścieżki)