Programowanie obiektowe – zadanie oceniane (3/4)

28-05-2018

Należy wysłać rozwiązanie w 90 minut po rozpoczęciu.
Liczba punktów do zdobycia: 25.

1. Wysyłanie rozwiązania

Należy spakować projekt w archiwum zip i nadać mu nazwę: login.zip
Należy wysłać email:
To: bednarzm@student.mini.pw.edu.pl
Subject: [PO] Zadanie3/4 2018
Załączniki: login_studenta.zip

2. Opis ogólny

Powstała pilna potrzeba stworzenia dużej liczby fikcyjnych personaliów (imię, nazwisko, itp.), powstalych z losowych kombinacji pewnej liczby imion i nazwisk. Klient chce mieć aplikację, która automatycznie wygeneruje dowolną ilość takich danych.

3. Opis szczegółowy

Część A (16 pkt).
Celem tego etapu jest stworzenie klas, które wygenerują nam pewną liczbę osób mających imiona i nazwiska wzięte z plików. Następnie osoby te zostaną pogrupowane po wykrytej płci i zapisane do dwóch plików.

Objaśnienie zawartości pliku nazwiska:
Wiśniewska;0,65;
Wójcik;0,62;
Kowalczyk;0,62;

[Nazwisko];[Parameter popularności];

Powinny być stworzone co najmniej wymienione poniżej obiekty.

- Person
  Klasa ma mieć konstruktor, który tworzy obiekt na podstawie wczytanej linijki (parsuje elementy), oraz nadpisaną metodę toString(), która zawiera elementy klasy zrzucone to tekstu. Pola:
  - Name
  - Surname
  - sex
  - age

- Name
  Klasa zawiera imię lub imiona. Powinna ona mieć konstruktor, który tworzy obiekt na podstawie wczytanej linijki (parsuje elementy), oraz nadpisaną metodę toString(), która zawiera elementy klasy zrzucone to tekstu. Pola:
  - String firstName
  - String secondName

- Surname
  Klasa zawiera imię lub imiona
  - String surname
  - double surnamePopularity

Akcja:
- Należy wczytać pliki z imionami i nazwiskami i wgrać je do klas poprzez konstruktory.
- Na podstawie tych danych należy wygenerować 10 000 osób o kombinowanych personaliach (losowane imię x losowane nazwisko), przy czym
  - Do osoby należy wylosować jeszcze wiek z przedziału 18-25.
  - Jeżeli imię kończy się na "a", to zakładamy, że jest to imię żeńskie. Dobrane nazwisko nie może być męskie. Też musi się kończyć na "a".
  - Imię "męskie" (które nie kończy się na "a") nie może mieć nazwiska żeńskiego, które kończy się na "a". Powinno kończyć się inną literką.
  - Do osoby należy też dopisać ustaloną w w/w sposób płeć.
- Obiekty grupujemy po wieku.
- Dla każdego wieku tworzymy plik ludzie_wiek.txt i wpisujemy tam wszystkich ludzi w daych wieku. Przykładowy wiersz to: Anna Joanna, Kowalska, wiek: 19, płeć: kobieta

Wymagania:

- Należy pamiętać, że imiona były kodowane w UTF-8, a nazwiska w ISO-8859-2
- Niektóre imiona są podwójne, trzeba o tym pamiętać
- Operacje wczytywania imion i nazwisk, generowania i zapisu powinny być oddzielne (oddzielne metody)
- Powinien być stworzony interfejs "Generator" i implementująca go klasa z metodami do:
  - wczytywania danych z pliku (metoda "wczytajImiona" zwracająca coś, co przechowuje wczytane imiona)
  - wczytywania danych z pliku (metoda "wczytajNazwiska" zwracająca coś, co przechowuje wczytane nazwiska)
  - generowania ludzi (metoda generuj(lista imion, lista nazwisk))
  - grupowania obiektów (metoda "grupuj", zwracająca coś co przechowuje pogrupowaną względem wieku strukturę obiektów)
  - zapisu podgrupowanych obiektów do plików (pobiera argument zwrócony przez poprzednią metodę)
- Należy założyć, że co któryś wiersz może być pusty (linijka zwraca np. "")
- Przy okazji zapisu do pliku, na konsoli powinna zostać wyświetlona informacja: o liczbie wygenerowanych ludzi w danym wieku.
- Zapis ma być dokonany w kodowaniu Cp1250

Część B (5 pkt).
Celem niniejszego fragmentu jest umożliwienie serializacji i deserializacji wycztaych elementów oraz uwrażliwienie algorytmu na dodatkowe okoliczności, a także wprowadzenie

Wymagania:

- Interfejs "Generator" zyskuje nowe metody
  - serializuj – zapisuje całą strukturę obiektów do pliku
  - deserializuj – wczytuje całą strukturę obiektów do pliku
- Separatorem w nazwiskach może być nie tylko ";" ale również i "+" oraz "-"
- Osoby zapisane do danego pliku (z w ramach danego wieku), mają być posortowane względem nazwiska, a jak się trafią takie same, to patrzymy również względem imienia.

Część C (4 pkt).
Udziwnienia.

Należy rozszerzyć klasę java.io.BufferedWriter o metodę Person writePerson(Person), która w argumencie przyjmuje instancję obiektu Person i użyć jej w łańcuszku wywołań. Należy dodać też metodę, która wypisuje na konsoli liczbę zapisanych do pliku kobiet poniżej 21 roku życia.

Nie można dopuścić, żeby wygenerowane osoby się powtarzały, gdzie definicją tego, że dwie istoty są takie same jest: identyczne imię, nazwisko i wiek. Powtarzające się jednostki należy wypisywać na konsoli.