4. Pliki i dane

Zadanie 4.1 Dane z pliku "sprzedaż"

W pliku sprzedaz.csv znajdują się dane opisujące sprzedaż różnych sklepów - w każdej linii jest jedna transakcja. Cena jest ceną jednostkową, w osobnej kolumnie jest podana ilość sztuk, a dopiero ich iloczyn jest wartością transakcji.

Utwórz odpowiednie klasy i napisz programy, które czytają dane z tego pliku i obliczają:

1. Sumę wartości wszystkich transakcji z całego pliku.
2. Ilość, sumę oraz minimalną, maksymalną i średnią wartość transakcji z wybranego miasta (niech program pyta o nazwę miasta na początku).
3. Dla każdego miasta występującego w pliku – sumę transakcji z tego miasta (wypisać bez powtórzeń), czyli zastosować schemat grupowania.

Możliwe sposoby, aby zrobić coś bardziej atrakcyjnie / nowocześnie:

- Zamiast zwykłej klasy utwórz rekord. Zastanów się czy warto zdefiniować tam jakąś własną metodę.
- Zadania rozwiąż zarówno w stylu „proceduralnym” w oparciu o pętle i ify, jak i w stylu „funkcyjnym” w oparciu o streamy, lambdy, collectory.
- Oprócz zadanych zadań, spróbuj znaleźć lub wyliczyć inne dane, np. w oparciu o datę, kategorię towaru, ...

Zadanie 4.2 Pan Tadeusz - Policz wybrane słowo w całym pliku

Napisz program, który obliczy ile razy dane słowo występuje w pliku
(np. Tadeusz w pliku pan_tadeusz.txt).

Podpowiedź:

Najlepiej użyć klasy Scanner bezpośrednio na pliku i użyć odpowiedniego wyrażenia regularnego opisującego separator pomiędzy słowami. Najlepszy sposób to:

```
Scanner sc = new Scanner(new File("pan_tadeusz.txt"));
sc.useDelimiter("[^\\p{L}]+");
while(sc.hasNext()) {
     String slowo = sc.next();
     // TODO co zrobić z każdym słowem
}
```

Wersja interaktywna

Użyj JOptionPane do pobrania szukanego słowa i wyświetlenia wyniku.

Użyj JFileChooser do wybrania pliku z dysku (showOpenDialog i inne kroki - poszukajcie "w internetach" jak się tego używa...).

Zadanie 4.3 Policz wszystkie słowa

Dla pliku takiego jak pan_tadeusz.txt (lub dowolny inny plik tekstowy) napisz program, który czyta słowa z tego pliku (schemat przedstawiony w zadaniu 3.3 oraz na zajęciach) i liczy, ile razy występuje każde słowo. Dla czytelności wyniki można wypisać posortowane alfabetycznie (to łatwiejsze) lub według liczby wystąpień (to trudniejsze, ale też się da).

Przykładowy fragment prawdziwych wyników:

```
ja              ->  101
jako            ->  105
dla             ->  106
Tadeusz         ->  107
ich             ->  108
pod             ->  108
gdy             ->  109
tylko           ->  113
Lecz            ->  114
```

Zadanie 4.4 Odmiana słów

Pod adresem **zil.ipipan.waw.pl/PoliMorf** znajdują się zasoby dotyczące odmiany wyrazów w języku polskim. Plik `PoliMorf-0.6.7.tab` (lub nowsza wersja) zawiera relację między słowem (pierwsza kolumna) a jego formą podstawową (druga kolumna).

Stwórz klasę, której obiekt zawiera odpowiednio przygotowaną „bazę wiedzy” nt. odmiany słów. Obiekt tej klasy tworzy się na podstawie pliku takiego jak `PoliMorf-0.6.7.tab` . Obiekt powinien umożliwiać znalezienie formy podstawowej (lub kilku kandydatów) dla podanego słowa odmienionego oraz odczytanie zbioru słów odmienionych na podstawie formy podstawowej.

Przykładowe użycie tej klasy mogłoby wyglądać tak (ale nazwy możecie wymyślić inne):

BazaOdmiany baza = BazaOdmiany.wczytaj("PoliMorf-0.6.7.tab");

Set<String> odmiany = baza.znajdzOdmiany("Tadeusz"); // [Tadeusz, Tadeuszowi, ...]

String podstawowa = baza.znajdzFormePodstawowa("Tadeusza"); // Tadeusz

Ten sam wyraz może być formą odmienioną dla kilku form bazowych, np. „dam” może pochodzić od „dać”, ale również od „dama”... Dlatego lepszym, bardziej ogólnym rozwiązaniem byłoby:

Set<String> podstawowe = baza.znajdzFormyPodstawowe("Tadeusza"); // [Tadeusz]

Korzystając z tej klasy napisz program, który w pętli odczytuje od użytkownika kolejne słowa i dla podanego słowa wyświetla jego formę podstawową lub informację, że nie znaleziono.

Wskazówka nt wydajności: Przyjmijcie, że budowanie bazy wiedzy w pamięci na podstawie pliku może chwilę potrwać (ale raczej sekundy niż minuty) i zająć trochę pamięci, natomiast szukanie form bazowych i odmian ma już działać szybko (ułamki sekund).

Zadanie 4.5 Policz słowa z odmianą

Wykorzystując klasę do odmiany słów, policz słowa sprowadzone do swojej formy podstawowej (pierwszej, jeśli jest kilka kandydatur).

Przykładowo zamiast całej serii odmienionych Tadeuszów

```
Tadeusz          →  107
Tadeusza         →   54
Tadeuszem        →    2
Tadeuszowi       →    5
Tadeuszu         →    6
```

powinien być jeden wpis

```
Tadeusz → 174
```

Zadanie 4.6 Liczba słownie

Stwórz klasę (+ ewentualnie klasy pomocnicze w tym samym pakiecie), w której zdefiniowana będzie funkcja zamieniająca podaną liczbę całkowitą na postać słowną. Przykładowe użycie:

String slownie = LiczbaSlownie.liczbaSlownie(113);

// "sto trzynaście"

Obsłuż jak największy zakres liczbowy, może cały zakres typu long?

Opcjonalnie dodaj także drugą funkcję zwracającą tekst mówiący o kwocie pieniężnej, tak jak umieszcza się ją np. na umowach czy fakturach, np.:

String kwota = LiczbaSlownie.kwotaSlownie(204);

// "dwieście cztery złote"

Opcjonalnie - obsługa „groszy”, wersje w innych językach i inne własne rozszerzenia.

Napisz testy jednostkowe oraz program interaktywny (może być okienkowy w Swing).

5. Aplikacje okienkowe (Swing)

Stwórz jedną lub więcej aplikacji okienkowych w technologii Swing. Wygląd interfejsu przygotuj w edytorze wizualnym, np. w NetBeans (New JFrameForm), ewentualnie w Eclipse z doinstalowaną wtyczką Window Builder (New WindowBuilder > Swing Designer > Application Window) lub w IntelliJ.

Można stworzyć aplikację według własnego pomysłu (mile widziane), albo wybrać coś poniższych propozycji.

Zadanie 5.1 BMI (tylko, jeśli ktoś potrzebuje bardzo prostego zadania)

Człowiek podaje swój wzrost i wagę, a otrzymuje wyliczony współczynnik BMI i informację tekstową czy jest w normie, czy ma się odchudzać, czy raczej przytyć.

Niektóre źródła na temat BMI rozróżniają normy ze względu na wiek lub płeć. Opcjonalnie możesz w swoim programie uwzględniać także te informacje.

Zadanie 5.2 Koszt paliwa

Użytkownik wpisuje: ile średnio pali w trasie jego samochód (l/100 km), ile kosztuje paliwo (zł/l) oraz długość trasy w km. Program wylicza ile kosztuje taka trasa.

Zadanie 5.3 Konwerter jednostek

Napisz program, który służy do przeliczania wartości między różnymi systemami miar. Minimum to program, który przelicza jedną parę, np. mile na kilometry. Można spróbować zrobić bardziej rozbudowaną aplikację, np. po jednej stronie ekranu umieść pola na dane w jednostkach metrycznych (centymetry, metry, kilometry, kilogramy, Celcjusze), a z drugiej brytyjskich (cale, stopy, mile, funty, Fahrenheity), a za pomocą przycisków można przeliczać w jedną lub w drugą stronę. Własne pomysły na układ okna i sposób działania mile widziane.

Zadanie 5.4 Automat na monety

Zrealizuj przykład z automatem parkingowym jako aplikację okienkową. Użytkownik podaje liczbę godzin, za które płaci, automat wylicza opłatę, następnie „wrzuca się monety” (np. przyciski dla różnych monet), a automat odejmuje wrzucone monety od kwoty do zapłaty, na końcu „wydaje resztę”.

Zamiast automatu parkingowego można wymyślić np. automat biletowy (z biletami normalnymi i ulgowymi), z kawą, z biletami do ZOO (normalne, ulgowe, rodzinne – z listy do wyboru) itp.

Zadanie 5.5 Kółko i krzyżyk

Program, który pozwala przeprowadzić rozgrywkę w kółko i krzyżyk. Można np. w oknie rozmieścić 9 przycisków, w których będą odpowiednie symbole. Gdy użytkownik kliknie w przycisk, jest to traktowane jako ruch i symbol się zmienia. Program prawidłowe kliknięcia

traktuje na przemian jako ruchy jednego lub drugiego gracza. Program sam powinien rozpoznać kiedy dochodzi do wygranej lub kiedy wszystkie pola zostają zajęte i gra kończy się remisem.

To zadania można spróbować zrobić bez edytora graficznego, pisząc wszystko od zera i używając GridLayout do rozmieszczenia 9 guzików.

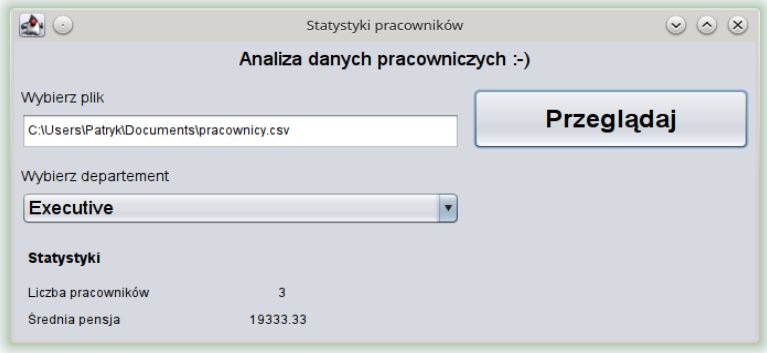
Zadanie 5.6 Wilk, koza, kapusta

Starożytna łamigłówka: Po jednej stronie rzeki znajdują się **wilk**, **koza** i **kapusta** oraz przewoźnik, który ma łodzią przewieźć je na drugą stronę rzeki. Problem polega na tym, że w łodzi zmieści się tylko jedno zwierzę/rzecz na raz – przewoźnik musi więc przewozić po jednej rzeczy i zostawiać je na brzegu. Jeśli jednak bez opieki pozostaną wilk i koza – wilk zje kozę; gdy zostaną zaś koza i kapusta – koza zje kapustę. Łamigłówka polega na tym, aby ustalić jak bezpiecznie przewieźć wszystkie trzy elementy na drugą stronę rzeki.

Zgodnie z opisanymi zasadami napisz prostą grę jako program w technologii Swing. W prostszej wersji możesz użyć wyłącznie przycisków i etykiet tekstowych. W miarę możliwości i dostępnego czasu możesz spróbować wprowadzić obrazki (poczytaj np. o ImageIcon i dodawaniu ich do JLabel/JButton).

Zadanie 5.7 Analiza danych z pliku CSV

Program ma umożliwiać analizę danych wczytywanych z pliku takiego jak emps.csv (ewentualnie można zrobić wersję dla zawodnicy.csv lub sprzedaz.csv...).



Po naciśnięciu przycisku "Przeglądaj" za pomocą JFileChooser można wybrać plik z dysku. Używając FileNameExtensionFilter wyświetlaj tylko pliki .csv. Po wybraniu pliku, jego dane powinny zostać wczytane (jako List<Employees>) i w odpowiednich labelach wyświetlone informacje sumaryczne: liczba pracowników, średnia pensja (ew. jeszcze min i max pensja).

Jeśli się uda: W oknie umieść JComboBox, z którego można wybrać nazwę departamentu. Lista departamentów (jako DefaultComboBoxModel) powinna być ustalona na podstawie danych z pliku! Po wybraniu departamentu z listy, w odp. miejscach okna wyświetl informację o ilości pracowników w tym departamencie oraz ich średniej pensji.