Zadania

W pliku data.zip dane są dwa katalogi: food oraz nonfood zawierające pliki CSV przedstawiające zmiany cen różnych produktów do stycznia 2010 do marca 2022 roku. Pliki zostały stworzone na podstawie danych dostępnych na stronach Głównego Urzędu Statystycznego. Każdy z plików w pierwszej linii zawiera nazwę produktu lub usługi. W drugiej linii znajduje się nagłówek tabeli. Nagłówek tabeli oraz jej wiersze różnią się od siebie w zależności od rodzaju pliku. Pliki z katalogu nonfood od pierwszej kolumny posiadają cenę produktu w danym miesiącu. Ceny produktów żywnościowych są rozróżniane ze względu na województwo. Pliki z katalogu food w pierwszej kolumnie posiadają województwo, a od drugiej kolumny pojawiają się ceny produktu w danym miesiącu.

Uwaga 1.
Przedstawione poniżej kroki stanowią propozycję kolejności rozwiązywania zadania. Po przeczytaniu całości można zdecydować o rozwiązywaniu w innej kolejności.

Uwaga 2.
Należy wysyłać wyłącznie pliki .java bez ich uprzedniego spakowania.

Uwaga 3.
Należy założyć, że pliki z danymi będą miały postać taką, jak powyżej opisana. Nie ma potrzeby sprawdzać ich poprawności.

Uwaga 4.
Należy założyć, że zapisy w plikach zaczynają się od stycznia 2010 roku, a każda kolejna kolumna dotyczy kolejnego miesiąca - nie trzeba parsować nagłówka tabeli.

Krok 0.
W pliku NonFoodProduct.java dana jest klasa **NonFoodProduct**, posiadająca prywatny konstruktor oraz dwa prywatne pola:

- **String name** - zawierający nazwę produktu,
- **Double[] prices** - tablica zawierająca ceny w kolejnych miesiącach od 01.2010.

W klasie zaimplementowany jest publiczny akcesor do pola name oraz metoda **fromCsv**, która odczytuje zawartość pliku CSV do takiej struktury. Zaznajom się z jej kodem.

Krok 1.
Napisz klasę abstrakcyjną **Product**. Niech po tej klasie dziedziczy istniejąca klasa **NonFoodProduct**. Przenieś do niej pole **name** pozostawiając je prywatnym. Przenieś także akcesor. W klasie **Product** zdefiniuj publiczną, abstrakcyjną metodę:

double getPrice(int year, int month).

Metoda powinna zwracać cenę produktu w danym roku i miesiącu, a jeżeli dane nie mieszczą się w zakresie 01.2010 - 03.2022, lub gdy miesiąc będzie spoza zakresu 1-12 rzucić wyjątek **IndexOutOfBoundsException**. Nadpisz i zaimplementuj tę metodę w klasie **NonFoodProduct**. Przetestuj jej działanie w metodzie **Main::main**.

Krok 2.
Napisz klasę **FoodProduct** dziedziczącą po **Product**. Stwórz w niej statyczną, publiczną metodę wytwórczą, analogiczną do tej istniejącej w **NonFoodProduct**:

FoodProduct fromCsv(Path path),

działającej tak, aby możliwe było wywołanie opisanych w dalszej części tego kroku metod.

Klasa powinna posiadać publiczną metodę:

double getPrice(int year, int month, String province).

Metoda ma zwracać cenę w określonym województwie przekazanym napisem składającym się z wielkich liter (jak w plikach z danymi). Jeżeli zostanie podany napis nie pasujący do żadnego województwa lub data będzie niewłaściwa, należy rzucić wyjątek **IndexOutOfBoundsException**.

Klasa powinna także nadpisywać metodę:

double getPrice(int year, int month)

w taki sposób, że jako wynik będzie zwracana średnia arytmetyczna cen ze wszystkich województw.

Przetestuj działanie dwu- i trójargumentowej metody **FoodProduct::getPrice** w metodzie **Main::main**.

Krok 3.
W klasie **Product** stwórz prywatną, statyczną listę obiektów klasy **Product**. Napisz statyczną, publiczną metodę **Product::clearProducts** czyszczącą listę **products** oraz metodę **Product::addProducts** dodającą do niej elementy, która przyjmie dwa parametry:

- obiekt funkcyjny, do którego można przypisać metody **FoodProduct::fromCsv** oraz **NonFoodProduct::fromCsv**,
- obiekt **Path** zawierający ścieżkę do katalogu z plikami danych.

Metoda **Product::addProducts** powinna dodać do obiektu **products** obiekty utworzone na podstawie plików z danymi.

W metodzie **Main::main** należy wywołać metodę **Product::addProducts** dwa razy: dla ścieżki *“data/nonfood”* i metody **NonFoodProduct::fromCsv** oraz dla ścieżki *“data/food”* i metody **FoodProduct::fromCsv**.

Krok 4.
Napisz klasę wyjątku **AmbigiousProductException**, która przyjmuje w konstruktorze obiekt **List<String>** i zaprogramuj ją tak, by wyświetlała jej zawartość w stosie błędów (formatowanie jest dowolne).

Napisz publiczną, statyczną metodę **Product::getProducts** rzucającą wyjątki **AmbigiousProductException** oraz **IndexOutOfBoundsException**. Metoda powinna przyjąć napis będący prefiksem nazwy produktu (zwracanej przez **Product::getName**). Jeżeli w liście produktów znalezione zostanie:

- 0 produktów o nazwie rozpoczynającej się od tego prefiksu - należy rzucić wyjątek **IndexOutOfBoundsException**, np. prefiks *"Abc"* nie wskazuje na żaden obiekt,
- 1 produkt - należy go zwrócić, np. prefiks *"Bu"* wskazuje jednoznacznie na obiekt, którego metoda **Product::getName** zwróci: *"Buraki - za 1 kg"*,
- więcej niż 1 produkt - należy rzucić wyjątek **AmbigiousProductException**, przekazując mu jako parametr konstruktora, listę nazw przedmiotów zaczynających się od danego prefiksu, np. prefiks *"Ja"* powinien spowodować rzucenie wyjątku, któremu jako parametr konstruktora zostanie przekazana lista: [*"Jabłka - za 1 kg"*, *"Jaja kurze świeże - za 1 szt"*].

W metodzie **Main::main** przetestuj wszystkie trzy wymienione przypadki.

Krok 5.
Napisz klasę **Cart** posiadającą publiczne metody

- **void addProduct(Product product, int amount)** - dodającą do koszyka produkt w liczbie sztuk określonej zmienną **amount**,
- **double getPrice(int year, int month)** - zwracającą wartość koszyka w zł we wskazanym roku i miesiącu,
- **double getInflation(int year1, int month1, int year2, int month2)** - zwraca procentową wartość inflacji w ujęciu rocznym między dwoma wskazanymi miesiącami na podstawie zawartości koszyka, zakładając, że y1, m1 < y2, m2. Należy ją wyliczyć według wzoru:

(price2 - price1) / price1 * 100 / months * 12,
gdzie price1 i price2 to wartości koszyków w dwóch wskazanych miesiącach, a months to liczba miesięcy dzieląca wskazane daty.

W metodzie **Main::main**, przy użyciu metody **Product::addProduct** dodaj do koszyka kilka produktów i wywołaj na jego rzecz metody **Cart::getPrice** i **Cart::getInflation**.

Zadanie 1a.

Wydziel metodę obliczającą indeks tablicy na podstawie miesiąca i daty do statycznej, publicznej metody w klasie Product.

Zadanie 1b.
Napisz test sprawdzający poprawność działania przeliczania indeksu.

Zadanie 1c.
Napisz test sprawdzający poprawność rzucania wyjątku IndexOutOfBoundsException.

Zadanie 1d.
Napisz parametryzowany test sprawdzający poprawność działania przeliczania indeksu.

Zadanie 1e.
Przygotuj plik CSV zawierający wszystkie potencjalne, miesiące, lata i odpowiadające im indeksy z przedziału 01.2010 - 03.2022. Napisz test wykorzystujący ten plik.

Zadanie 1f.
Napisz test sprawdzający wszystkie potencjalne miesiące od 2001 roku. Uczyń asumpcję, która spowoduje, że test dla lat wcześniejszych niż 2010 nie będzie sprawdzany.

Zadanie 2a.
Napisz test sprawdzający poprawność załadowania danych z pliku przez metodę FoodProduct::fromCsv na podstawie nazwy odczytanej z obiektu za pomocą akcesora.

Zadanie 2b.
Napisz test sprawdzający poprawność załadowania danych z pliku przez metodę FoodProduct::fromCsv na poprzez trójargumentową metodę Product::getPrice.

Zadanie 2c.
Napisz test sprawdzający poprawność załadowania danych z pliku przez metodę FoodProduct::fromCsv na poprzez dostęp do mapy cen będącej prywatną zmienną.

Zadanie 2d.
Napisz test sprawdzający poprawność załadowania danych z pliku przez metodę FoodProduct::fromCsv na poprzez dostęp do mapy eksponowanej przez akcesor zdefiniowany w klasie dziedziczącej po FoodProduct. Wprowadź konieczne, minimalne zmiany w kodzie FoodProduct.

Zadanie 3a.
W celu przetestowania metody Product::getProduct załóżmy, że metoda Product::addProducts jest już przetestowana. Napisz metodę uruchamianą przed testami, która:

- utworzy tymczasowy katalog,
- utworzy w niej kilka plików o strukturze odpowiadającej plikom oczekiwanym przez FoodProduct::fromCsv,
- doda je za pomocą metody Product::addProducts do statycznej listy

Napisz metodę, która wywoła się po wszystkich testach, która usunie tymczasowy katalog wraz zawartością.

Napisz test sprawdzający odczytywanie prefiksu w metodzie Product::getProduct poprzez porównanie go z nazwą znalezionego obiektu.

Zadanie 3b.
Napisz fabrykę dynamicznych testów, które przetestują poprawność działania metody Product::getProduct dla wszystkich wygenerowanych plików.