Projekt indywidualny
Specyfikacja Implementacyjna

Arkadiusz Michalak

Algorytmy i struktury danych
10.11.2018

Spis treści

1 Omówienia metod

Na stronie trzeciej znajduje się diagram klas. Podzielonych na pakiety. Omówione zostaną teraz poszczególne klasy

1.1 Main Class

Klasa głowna zawierająca metodę main, będzie odpowiadać za przyjęcie argumentów podanych wsadomo. Następnie względem nich będzie prowadzone sterowanie. Klasa ta ma dostęp do metod publicznych klasy Load i klasy Graph.

1.2 Graph Package

Główny pakiet programu, zawiera klasy Graph, Vertex oraz Rate.

1.2.1 Graph

Klasa tworząca graf i sterująca jego przetwarzaniem. Zawiera listę wierzchołków grafu. Metoda addVertex służy dodawaniu nowego wierzchołka, jak argument przyjmuje kod waluty który stanowi nazwą wierzchołka. Metoda addRate dodaje do odpowiedniego wierzchołka dane kursu względem innego wierzchołka. Należy sprawdzić czy wierzchołki o takich nazwach istnieją. Dwie metody dodające będą wywoływane przez klase Load pobierające dane z pliku wejściowego. Metoday getBestExchange i getArbitrag zwracają listę wierzchołków którę stanowią odpowienio najlepszą wymiane waluty wejsciowej na wyjściową lub dowolny arbitraż. W celu zwrócenia tych danych klasa Graph wywoła metodę checkGraph, punktowy przebieg tego jak ma zadziałać zawarty w niej algorytm przedstawiono w rozdziale 2.Opis algorytmu. Po przetworzeniu grafu odczyt optymalnej ścieżki odbywa się przy użyciu metody readBestRoad.

1.2.2 Vertex

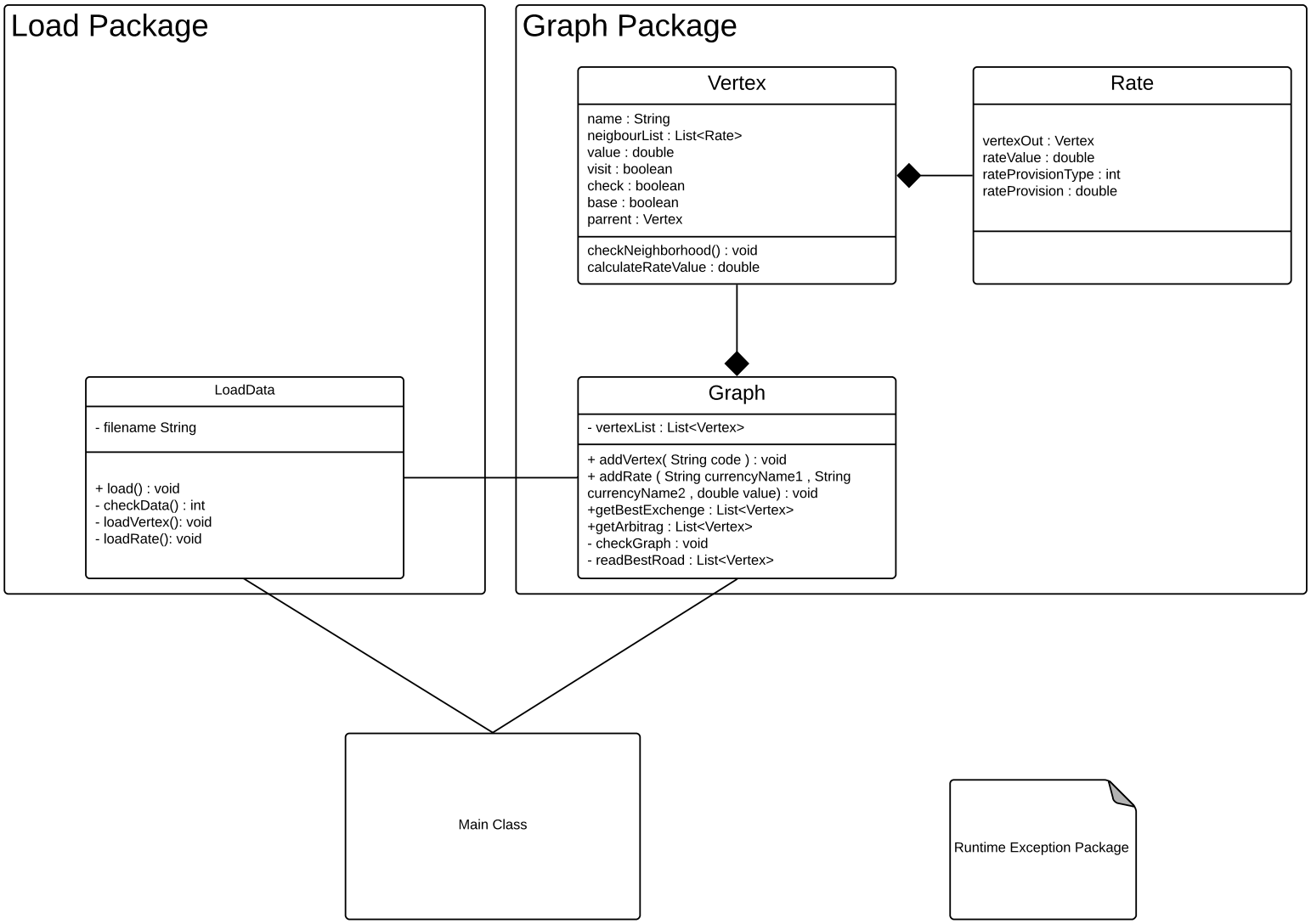
Klasa wierzchołek, wszystkie jej pola i metody są domyślnego czyli pakietowego dostępu. Zawiera metodę checkNeighborhood, która przechodzi listę sąsiadów danego wierzchołka. Druga metoda calculateExchenge służy do obliczenia ilości waluty wyjściowej po wymianie.

1.2.3 Rate

Klasa ta opisuje krawędzie grafu to jest połączenia miedzy walutami. Ma charakter typowo pomocniczy, wprowadzono ją w celu uproszczenia i wyeksponowania ważnych danych wymiany.

1.3 Load

Klasa znajduje się w swoim własnym pakiecie. Jej zadaniem jest odczytania danych z pliku i po sprawdzeniu ich poprawności przekazanie ich do klasy Graph która utworzy odpowiednie, wyżej opisane struktury. Metoda publiczna load wywołana jest przez klase sterującą, odczytuje ona cały plik, wcześniej sprawdzając jego poprawność metodą checkData.


Load Package
Graph Package
Vertex
name : String
neigbourList : List<Rate>
value : double
visit : boolean
check : boolean
base : boolean
parrent : Vertex
checkNeighborhood() : void
calculateRateValue : double
Rate
vertexOut : Vertex
rateValue : double
rateProvisionType : int
rateProvision : double
LoadData
- filename String
+ load() : void
- checkData() : int
- loadVertex(): void
- loadRate(): void
Graph
- vertexList : List<Vertex>
+ addVertex(String code) : void
+ addRate (String currencyName1 , String currencyName2 , double value) : void
+getBestExchenge : List<Vertex>
+getArbitrag : List<Vertex>
- checkGraph : void
- readBestRoad : List<Vertex>
Main Class
Runtime Exception Package

2 Opis algorytmu

Najistotniejsza część programu czyli algorytm przeszukujący graf znajdująca się w klasie "" będzie działał według poniższego opisu:

1. Działanie rozpoczynamy w wierzchołku zadanej waluty, wstawiając zadaną wartość - suma którą posiadamy oraz ustawiając w nim że jest wierzchołkiem wyjściowym.
2. Przechodzimy do sąsiedniego wierzchołka.
3. Ustalamy wartość jego waluty na podstawie kursu i prowizji.
4. Jeśli ustalona wartość jest wyższa niż obecna wartość wierzchołka to ustawiamy ją jako nową wartość wierzchołka i zmieniamy jego przodka na nasz wierzchołek bazowy.
5. Jeżeli to węzeł wyjściowy i szukamy arbitrażu to wychodzimy z algorytmu.
6. Ustawiamy go jako odwiedzony.
7. Kroki 3-6 powtarzamy dla wszystkich sąsiadów.
8. Ustawiamy w wierzchołka bazowym, że został przeszukany.
9. Kroki 1-8 powtarzamy dla wszystkich wierzchołków grafu.

Po wykonaniu powyższych działań znajdujemy ukrytą w grafie ścieżkę arbitrażu najkorzystniejszej wymiany. Dla arbitrażu zaczynamy od wierzchołka waluty wejściowej, dla wymiany wierzchołka waluty wyjściowej i wykonujemy następujące kroki:

1. Zapisujemy kod waluty wierzchołka do listy.
2. Przechodzimy do przodka wierzchołka.
3. Kroki 1-2 wykonujemy aż dotrzemy do wierzchołka waluty wejściowej.

3 Scenariusze testów

W poniższym rozdziale krótko zostaną omówione testy poszczególnych metod publicznych.

4 Informacje o sprzęcie

Program zostanie napisany w języku Java, wersji: 1.8.0. Używając systemu operacyjnego Windows 10 wersji 64-bitowej. Komputer na którym program będzie testowany posiada:

- procesor: Intel Core i7-4700HQ 2.40GHz,
- pamięć ram: 8 GB DDR3 1600MHz,
- procesor graficzny: Intel HD Graphics 4600.