Zadanie nr 2 z APRO2

Tomasz Mycielski (304248)

TO SPRAWOZDANIE MA 637 SŁÓW, A JEGO PRZECZYTANIE ZAJMUJE OKOŁO 5 MINUT.

Zbiór danych

Jako zbiór danych wybrałem spis transakcji sprzedaży na rynku nieruchomości w hrabstwie Sacramento w Kalifornii w trzecim tygodniu maja 2008. Każdy wiersz pliku opisuje jedną transakcję. Plik `Sacramentorealestatetransactions.csv`, który przetwarzam ma 12 kolumn danych:

- street
- city
- zip
- state
- beds
- baths
- sq__ft
- type
- sale_date
- price
- latitude
- longitude

W pliku tym opisane są 942 transakcje.

Klasa `CSVReader`

Do wczytania pliku utworzona została klasa `CSVReader`. Obiekt tej klasy bierze jako parametr konstruktora ścieżkę do pliku. Pierwszy wiersz pliku odczytywany jest jako nagłówek, a wartości jego pól danych zapisywane są do tablicy `String[] headerRow`.

Metoda `streamValues()` w klasie `CSVReader` zwraca strumień wartości wszystkich pól danych z pliku csv. Wartości odczytywane są za pomocą klasy `Scanner` z wykorzystaniem znaku `,` jako ogranicznika:

```
        Scanner scanner = new Scanner(new File(filepath)); // filepath to ścieżka do
pliku
        scanner.useDelimiter(",");
```

Wartości odczytywane wiersz po wierszu dodawane są do stream buildera, aby po odczytaniu całego pliku zostały zwrócone w formie strumienia.

```
        Stream.Builder<String> builder = Stream.builder();
        while (scanner.hasNextLine()) {
            for (String string :
                scanner.nextLine().split(",")) {
                builder.add(string);
            }
        }
        return builder.build();
```

Metoda `toString()` zwraca informację o zawartości pierwszego wiersza pliku (nagłówka) oraz o liczbie wierszy w pliku.

Ponadto w klasie są jeszcze metody `getColumns()` i `countLines()`, które robią dokładnie to, na co wskazuje ich nazwa.

Klasa `Main`

Po wczytaniu pliku i wypisaniu na konsolę podstawowych informacji o nim wartości jego pól danych zapisywane są ze strumienia do ArrayListy `list`:

```
ArrayList<String> list = (ArrayList<String>)
csvReader.streamValues().collect(Collectors.toList());
```

W takim stanie lista nie nadaje się do wykonywania na niej operacji z danymi. Należy najpierw usunąć pierwsze dwanaście pól danych, czyli wiersz nagłówkowy pliku:

```
list.subList(0, csvReader.getColumns()).clear();
```

Następnie z kolumny `city` wszystkie wartości zapisuję do zbioru `set`. Dzięki temu będę miał wszystkie miasta zapisane w jednej strukturze danych bez powtórzeń.

Aby obliczyć statystyczne dane dla każdego z miast, tworzę HashMapę `HashMap<String, int[]> pricesHashMap`. Jej kluczem będą nazwy miast, a wartością tablica trzech integerów (kolejno):

- sumie kosztu wszystkich zakupionych nieruchomości,
- ilości transakcji w tym mieście,
- sumie stóp kwadratowych wszystkich zakupionych nieruchomości.

Ceny w pliku podane są jako liczby całkowite, można więc korzystać z typu `int` do przedstawiania ich w programie.

W celu uzyskania interesujących nas danych z pliku csv i zapisania ich do HashMapy wykorzystana została pętla `for`:

```
        for (int i = 1; i < list.size() - 8; i += 12) {
            int price = pricesHashMap.get(list.get(i))[0];
            int denominator = pricesHashMap.get(list.get(i))[1];
            int squareFootage = pricesHashMap.get(list.get(i))[2];
            squareFootage += Integer.parseInt(list.get(i + 5));
            price += Double.parseDouble(list.get(i + 8));
            denominator++;
            pricesHashMap.put(list.get(i), new int[]{price, denominator,
squareFootage});
        }
```

Nazwa miasta przechowywana jest w pliku csv kolumnie o indeksie `1`, powierzchnia w kolumnie `6`, cena w kolumnie `9`. Jeśli pod indeksem `i` w `list` zapisana jest nazwa miasta, to pod indeksem `i+5` będzie powierzchnia sprzedanej nieruchomości, a pod indeksem `i+8` będzie jej cena; możemy zatem zapisać wartości z komórki `i+5` i `i+8` parsować z typu `String` na typ `int`.

Dla każdej transakcji jej dane zapisywane są do `pricesHashMap`:

- koszt transakcji jest dodawany do sumy pieniędzy wydanych na nieruchomości w tym mieście,
- licznik transakcji w tym mieście zwiększany jest o 1,
- suma powierzchni zakupionych nieruchomości w tym mieście jest zwiększana o powierzchnię nieruchomości, której dotyczyła ta transakcja.

Po wykonaniu tych operacji dla każdej transakcji obliczana jest średnia cena nieruchomości dla każdego miasta, średnia cena jednej stopy kwadratowej dla każdego miasta oraz odchylenie standardowe ceny jednej stopy kwadratowej.

Do wykonania tych obliczeń wykorzystywane są metody `mean(LinkedList<Integer> listOfIntegers)` oraz `stDev(LinkedList<Integer> listOfIntegers, int mean)`. Metoda `stDev` do obliczenia sumy wykorzystuje strumień:

```
double summation = listOfIntegers.stream().mapToDouble(integer -> Math.pow((double) integer - (double) mean, 2)).sum();
```

Na podstawie obliczonych danych dla każdego z miast na konsolę wypisane zostają miasta, w których ceny nieruchomości są najwyższe. Ponadto na podstawie odchylenia standardowego wypisany zostaje na konsolę wniosek, czy różnice w cenach nieruchomości w regionie są wysokie, czy niskie, w zależności od miasta ich zakupu.