15.09.2017

Teoria Przestrzeni Danych i Algorytmów

**Detekcja i naprawa nieprawidłowych danych paliwowych**

**Raport z implementacji**

Autorzy:

Łukasz Janas

Dominik Popanda

Semestr 1, Grupa ITS1, Rok akademicki 2016/2017

# Wstęp

Celem pracy było opracowanie i implementacja mechanizmu wykrywającego różnego rodzaju nieprawidłowości i błędy w danych paliwowych oraz umożliwiającego ich naprawę. Dane te pochodzą ze stacji paliw i zawierają informacje dotyczące objętości przechowywanych paliw płynnych oraz ilości sprzedanego i dostarczonego paliwa.

Aplikacja miała wykrywać błędy w zestawach danych jak np.: brakujące lub niepełne rekordy, błędne wartości (znacząco odstające od trendu), dane w nieprawidłowej kolejności powstałe na skutek opóźnień itd. oraz również podejmować próbę ich naprawy lub w ostateczności usunięcia.

# Opis implementacji oraz analizy działania

Aplikacja została napisana w języku Java w wersji 1.8. Dodatkowo do walidacji danych użyto biblioteki Apache Commons Validator 1.6. Aplikacja jest uruchomiana z konsoli.

Początkowo utworzone 3 podstawowe klasy: NozzleMeasure, Refuel, TankMeasure, mające przechowywać dane z plików: nozzleMeasures.log, refuel.log, tankMeasures.log. Klasy te mają pola odpowiadające opisowi zawartemu w dokumencie Format danych.docx

Naszą aplikację możemy podzielić na 2 mniejsze aplikacje uruchamiane z klas o nazwie:

BreakMain

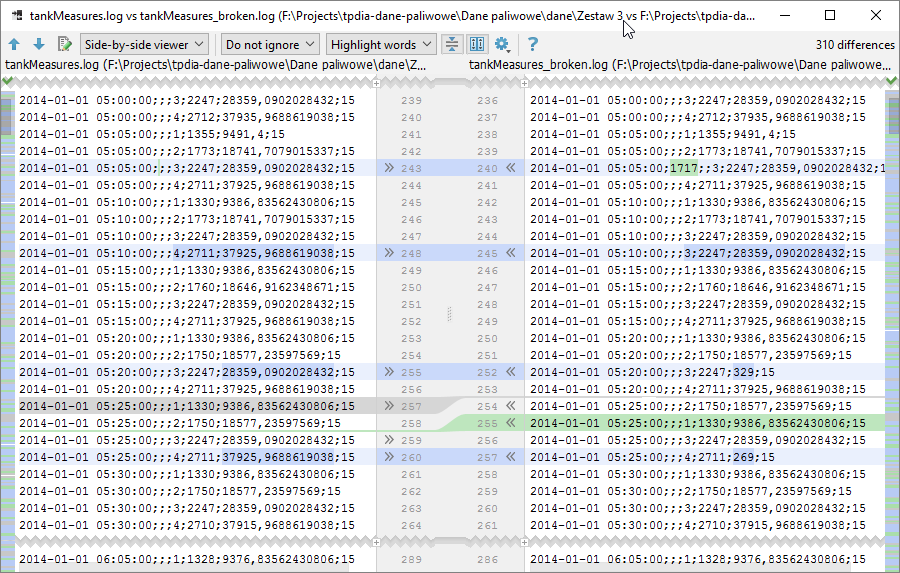
Moduł „psujący” rekordy (rekord - jedna linia pliku z danych wejściowymi).

Wczytuje rekordy z plików znajdujących się w folderach dane/Zestaw 1-3. Następnie tworzy nieprawidłowości w danych poprzez wywoływanie na rekordzie kilku metod, które znajdują się w klasie DataBreaker:

* remove() – usuwa cały rekord z pliku
* duplicate() – podwaja rekord w pliku
* swapLines() – zamienia kolejność 2 sąsiadujących linii w pliku
* attributesBreaker.run() – wywołuje metody psujące pojedyncze atrybuty wewnątrz jednego rekordu: usuwa atrybut, zamienia kolejność, wstawia zero.

Wszystkie operacje są wykonywane losowo, przy użyciu generatora pseudolosowego.

Pliki następnie zapisywane są w odpowiednich folderach dane\_broken/Zestaw 1-3



Rysunek przykład pliku z dodanymi nieprawidłowościami (po lewej plik oryginalny, po prawej popsuty)

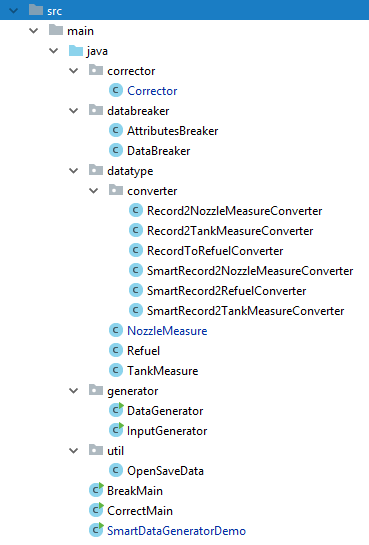
CorrectMain

Moduł naprawiający dane.

Wczytuje rekordy z plików znajdujących się w folderach dane\_broken/Zestaw 1-3. Dane są konwertowane do obiektów odpowiedniego typu przy użyciu konwerterów, np. SmartRecord2NozzleMeasureConverter. Można zauważyć że w projekcie występuje jeszcze druga wersja konwerterów, np. Record2NozzleMeasureConverter. Jest to pierwotna wersja nieuwzględniająca walidacji poprawności rekordów pobieranych z plików. Następnie jest wykonywana próba naprawienia danych za pomocą metod znajdujących się w klasie Corrector:

* removeDuplicates() – usuwa powtarzające się linie z pliku
* sortList() – przywraca poprawną kolejność zamienionych wcześniej rekordów.

Dodatkowo, podczas wczytywania danych za pomocą konwerterów, sprawdzana jest ich poprawność przy użyciu biblioteki Apache Commons Validator.

Rekordy są następnie serializowane z powrotem do plików i zapisywane w folderach dane\_corrected/Zestaw 1-3

Rysunek Struktura projektu: klasy

# Uruchomienie

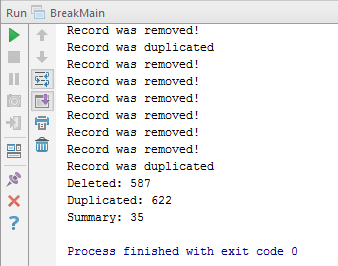
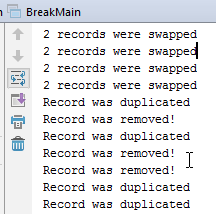
Moduły aplikacji uruchamiane są z konsoli. Należy podać następujące parametry:

BreakMain.jar <folder\_z\_plikami\_oryginalnymi> <folder\_z\_plikami\_popsutymi>

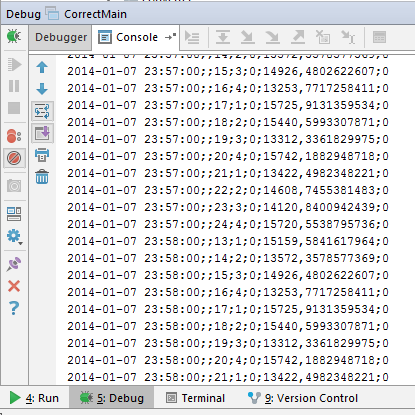
Program wyświetla informacje o wykonywanych operacjach („psucie” danych), a także podsumowanie ile rekordów zostało zmienionych.

CorrectMain.jar <folder\_z\_plikami\_popsutymi> <folder\_z\_plikami\_naprawionymi>

Program wyświetla informacje na temat wczytywanych oraz naprawionych rekordów.



Rysunek Przykładowe uruchomienia modułu BreakMain psującego dane



Rysunek Przykładowe uruchomienia modułu CorrectMain naprawiającego dane

# Wnioski

Stworzenie niniejszej aplikacji pozwoliło na praktyczną próbę rozwiązania problemu analizy oraz naprawy danych pomiarowych pochodzących z rzeczywistych systemów.

Podczas implementacji aplikacji pewne problemy sprawiały procesy deserializacji oraz serializacji danych - przetwarzanie danych z pliku do obiektów oraz z obiektów z powrotem do plików. Wymagało to wielokrotnej analizy wykonywanego kodu za pomocą debugera.

Kolejnym wyzwaniem było zaprojektowanie i implementacja metod pozwalających na tworzenie nieprawidłowych danych oraz ich naprawę.

Wszystkie problemy udało się jednak rozwiązać.