Specyfikacja funkcjonalna

1. Opis ogólny

   1.1. Nazwa programu
   Program nazywa się *VaccOpt*.

   1.2. Poruszany problem
   Program będzie optymalizował proces zakupu szczepionek przez hipotetyczne apteki.
   Każda z aptek będących pod opieką GM ma podpisany kontrakt z każdym dystrybutorem szczepionek leczniczych.
   Program napisany przeze mnie (jako członka ZA) ma za zadanie korzystając z danych zawartych w pliku dostarczonym przez GM ma znaleźć konfigurację, w której nastąpi minimalizacja kosztów sprzedaży szczepionek przy jednoczesnym zapewnieniu dostaw do wszystkich aptek.

   1.3. Użytkownik docelowy
   Przyjmując konwencję zadania użytkownikiem docelowym jest GM.
   Właściwym użytkownikiem docelowym naszego programu jest prowadzący zajęć laboratoryjnych.

2. Ogólna funkcjonalność

   2.1. Korzystanie z programu
   Program wykonany jest w formie aplikacji działającej w terminalu. Do włączenia go wymagane jest posiadanie kompilatora Java.

   2.2. Uruchamianie programu
   Aby włączyć program należy przejść w terminalu do katalogu zawierającego pilk wykonywalny **VaccOpt.jar** i włączyć go wpisując komendę *java -jar VaccOpt.jar ńazwaPliku"*, gdzie nazwaPliku to nazwa pliku z danymi. W przypadku wywołania bezargumentów program domyślnie przyjmuje korzyta z pliku *dane.txt*.

   2.3. Możliwości programu
   Program jest w stanie sprawdzić, czy przekazane mu *dane* zgadzają się z ustalonymi dyrektywami, na ich podstawie zoptymalizować, czyli w tym wypadku zminimalizować koszty zaopatrzenia

w pełni wszystkich aptek. Efektem końcowym działania programu jest wygenerowanie pliku *result.txt*.

3. Format danych i struktura plików

3.1. Słownik
GM - grupa menadżerska
ZA - zespół analityków
Klucz, *ID* - element unikatowy dla każdej apteki oraz dystrybutora
dane - w domyśle odpowiednio sformatowany plik dostarczony przez *GM* lub jego zawartość
GUI - graphical user interface, czyli interfejs graficzny widziany przez użytkownika

3.2. Struktura katalogów
Cały kod tworzący program będzie znajdował się w w katalogu *src*, tam też będzie plik main.java. Wszystkie testy znajdować się będą w katalogu *test*. Przykładowe *dane* będą znajdowały się w katalogu nadrzędnym dla*src* i *test*. Plik wyjściowy będzie generowany w tym samym folderze.

3.3. Przechowywanie danych w programie
Sam program znajduje się w repozytorium Wydziału Elektrycznego Politechniki Warszawskiej.
W programie informacje pobrane z danych będą przechowywane postaci tablic *kluczy*, z którymi będą powiązane pozostałe informacje dotyczące aptek i dystybutorów. Jedna tablica zawierać będzie *klucze* aptek, druga *klucze* dystrybutorów.

3.4. Dane wejściowe
Dane wejściowe znajdują się domyślnie w pliku dane.txt. Muszą zgadzać się one z przyjętym formatowaniem. Należy pamiętać, aby każda apteka i dystrybutor mieli *ID*. Dane w pliku muszą być oddzielone od siebie separatorem ", a nazwy aptek i dystrybutorów nie mogą zawierać tego znaku. Każda z aptek musi mieć połączenie z każdym z dystrybutorów. Wszystke liczby zawarte w pliku muszą być liczbami dodatnimi, a zapotrzebowanie na szczepionki jak i ich dostępna ilość muszą być liczbami całkowitymi. Przykładowy plik wejściowy:

```
# Producenci szczepionek (id | nazwa | dzienna produkcja)
0 | BioTech 2.0 | 900
1 | Eko Polska 2020 | 1300
2 | Post-Covid Sp. z o.o. | 1100
# Apteki (id | nazwa | dzienne zapotrzebowanie)
0 | CentMedEko Centrala | 450
1 | CentMedEko 24h | 690
2 | CentMedEko Nowogrodzka | 1200
# Połączenia  (id prod | id apt | max liczba szczepionek | koszt sztuki)
0 | 0 | 800 | 70.5
0 | 1 | 600 | 70
0 | 2 | 750 | 90.99
1 | 0 | 900 | 100
1 | 1 | 600 | 80
1 | 2 | 450 | 70
2 | 0 | 900 | 80
2 | 1 | 900 | 90
2 | 2 | 300 | 100
```

3.5. Dane wyjściowe

Dane wyjściowe znajdować się będą w pliku result.txt w poniższy sposób:

```
BioTech 2.0     -> CentMedEko Centrala [Koszt = 300 * 70.5 = 21150 zł]
Eko Polska 2020 -> CentMedEko Centrala [Koszt = 150 * 100 = 15000 zł]
/*
...
pozostałe ustalone połączenia pomiędzy producentami a aptekami
...
*/
Opłaty całkowite: 36150 zł
```

4. Scenariusz działania programu

4.1. Scenariusz ogólny

4.1.1. Włączenie programu

4.1.2. Sprawdzenie, czy podany plik jest odpowiednio sformatowany

4.1.3. Wyliczenie namniejszego możliwego kosztu

4.1.4. Zapisanie danych do pliku wyjściowego

4.1.5. Koniec działania programu

4.2. Scenariusz szczegółowy

i. Włączenie programu podając mu za argmunet własny plik z danymi, lub nie podając żadnego (wtedy pobierany jest plik domyślny)
ii. Sprawdzenie przez program, czy plik jest odpowiednio sformatowany. Podanie informacji o błędzie jeżeli taki nie jest.
iii. Zapisanie danych przez program w postaci tablic.
iv. Sprawdzenie, w której konfiguracji dla każdej z aptek będzie najkorzystniej kupić szczepionki.
v. Optymalizacja tego procesu.
vi. Sprawdzenie, czy można utworzyć plik wyjściowy i w przy pozytywnym rozpatrzeniu tego warunku utworzenie go. W przeciwnym wypadku komunikat błędu.
vii. Zapisanie wyniku w poprawnym formacie do pliku result.txt
viii. Zakończenie działania programu

5. Testowanie

I Ogólny przebieg testowania
Do testów kodu użyty zostanie JUnit 4. Testy będą miały na celu sprawdzenie jak się zachowuje program w wypadku podania nieprawidłowego pliku przez GM i czy zostanie zwrócona prawidłowa informacja o błędzie. Testy będą miały również sprawdzić, czy program zwraca prawidłowe dane i czy poprawnie tworzy nowe pliki wyjściowe.