Uber Heals - specyfikacja implementacyjna

Algorytmy i struktury danych

Paweł Cegielski, Piotr Szumański, Jakub Matłacz

data utworzenia: 3 grudnia 2020
data ostatniej zmiany: 14 grudnia 2020

1 Wstęp

Program udostępnia mapę obiektów, szpitali oraz dróg pomiędzy nimi. Pozwala wczytać listę pacjentów pliku lub przy pomocy interfejsu graficznego. Wizualizuje transport tychże osób do najbliższych niepełnych szpitali.

2 Opis ogólny

2.1 Nazwa programu

Nazwa programu to Uber Heals.

2.2 Poruszany problem

Problemem jest optymalna dystrybucja pacjentów, jednocześnie przestrzegając nałożonych ograniczeń takich jak istnienie dróg między lokacjami, ilość wolnych łóżek w szpitalach oraz położenie najbliższego szpitala od lokacji pacjenta.

2.3 Użytkownik docelowy

Program stworzono dla pracowników służby ochrony zdrowia odpowiedzialnych za transport pacjentów.

3 Opis implementacji

3.1 Użyta technologia

Program zrealizowano przy użyciu języka programowania Java w wersji 13 w paradygmacie programowania obiektowego. Aby go uruchomić należy posiadać zainstalowane Java Runtime Environment (JRE w wersji 8) oraz bibliotekę JUnit 4 i JavaFX do realizacji GUI.

3.2 Struktura programu

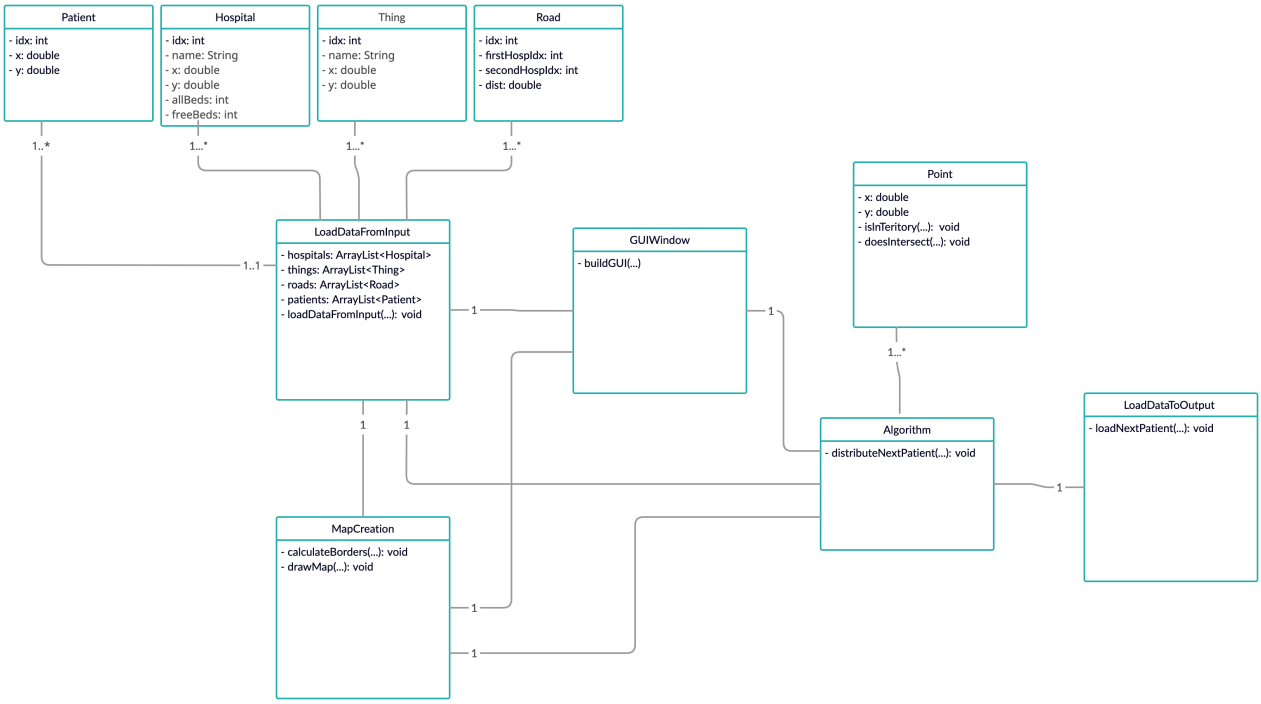
Program podzielono na klasy odpowiednialne za poszczególne zadania. Ponieżej (Rysunek 1) przedstawiono je wraz z ważniejszymi polami i metodami oraz asocjacje między nimi.

3.3 Sposób uruchamiania

Należy uruchomić plik .jar i zaczekać na pokazanie się okna interfejsu graficznego.

3.4 Struktury danych

W programie zastosowano ArrayList do przechowywania obiektów własnych klas oraz tablice, gdy potrzeba było większej szybkości.



Rysunek 1: Diagramy klas

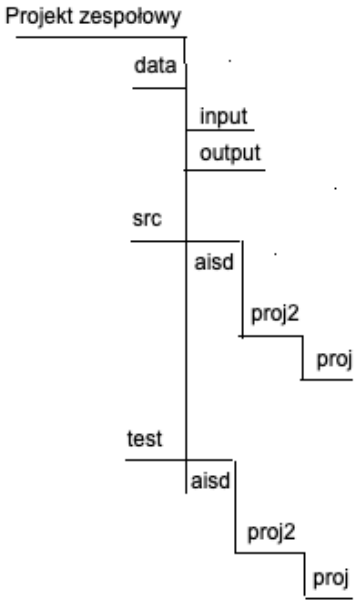
3.5 Struktura katalogów

Taka jak na Rysunku 2.

3.6 Wczytywanie pliku wejściowego

Jeden z dwóch możliwych plików wejściowych można wczytać za pomocą obiektu klasy LoadDataFromInput z właściwym parametrem trybu (tryb wczytywania mapy lub pacjentów) oraz nazwą odpowiedniego pliku.

- Sprawdzenie czy nazwa pliku jest właściwa
  - Znaki alfanumeryczne
  - Spacje, podłogi, myślniki
  - Duże i małe znaki
  - Rozszerzenie .txt
  - Brak znaku potoku, pustej nazwy "", nazwy będącej samym ".txt", innego rozszerzenia
- Sprawdzenie formatu pliku wejściowego
  - Właściwy format wierszy danych
  - Właściwy format nagłowków
  - Posiadanie wszystkich nagłowków
  - Oddzielanie wartości znakiem potoku
- Wczytanie danych do pól klasy we właściwej formie
- Sprawdzenie spójności danych wejściowych
  - Powtarzające się nazwy szpitali i obiektów
  - Powtarzające się id szpitali, obiektów, dróg i pacjentów



Rysunek 2: Struktura katalogów

  - Powtarzające się pary id szpitali w drogach

- Normalizacja identyfiaktorów dla dalszych obliczeń
- Przykładowy format plików wejściowych
  - Plik mapy

```
# Szpitale (id | nazwa | wsp. x | wsp. y | Liczba łóżek | Liczba wolnych łóżek)
1 | Szpital Wojewódzki nr 997 | 10 | 10 | 1000 | 100
2 | Krakowski Szpital Kliniczny | 100 | 120 | 999 | 99
3 | Pierwszy Szpital im. Prezesa RP | 120 | 130 | 99 | 0
4 | Drugi Szpital im. Naczelnika RP | 10 | 140 | 70 | 1
5 | Trzeci Szpital im. Króla RP | 140 | 10 | 996 | 0

# Obiekty (id | nazwa | wsp. x | wsp. y)
1 | Pomnik Wikipedii | -1 | 50
2 | Pomnik Fryderyka Chopina | 110 | 55
3 | Pomnik Anonimowego Przechodnia | 40 | 70

# Drogi (id | id_szpitala | id_szpitala | odległość)
1 | 1 | 2 | 700
2 | 1 | 4 | 550
3 | 1 | 5 | 800
4 | 2 | 3 | 300
5 | 2 | 4 | 550
6 | 3 | 5 | 600
7 | 4 | 5 | 750
```

  - Plik pacjentów

```
# Pacjenci (id | wsp. x | wsp.y)
1 | 20 | 20
2 | 99 | 105
3 | 23 | 40
```

3.7 Obliczanie dystrubucji pacjentów

Dla każdego następnego wczytanego pacjenta otwieramy pętlę while, której warunkiem stopu jest przydzielenie pacjenta do szpitala lub informacja o tym, że wszystkie szpitale są już pełne.

1. Wczytanie pacjenta.
2. Określenie czy należy do kraju, jeśli nie - koniec, przechodzimy do następnego pacjenta.
3. Przeniesienie do najbliższego, pod względem odległości euklidesowej, szpitala.
4. Przenoszenie pacjenta do kolejnego najbliższego (pod względem czasu podróży) szpitala tak długo, póki nie znajdzie się dla niego miejsce lub odwiedzi wszystkie możliwe szpitale i będą one pełne.
5. Jeśli pacjent zostanie przyjęty przez szpital to należy zmniejszyć ilość wolnych łóżek.

3.8 Tworzenie pliku wyjściowego

Plikiem wyjściowym naszego programu jest plik z logami, opisujący sekwencję zdarzeń - co dokładnie działo się z każdym pacjentem w kolejności. W logach przedstawiamy dla każdego pacjenta dokładną trasę jaką przebył. Widać z jakich do jakich współrzędnych się przemieścił oraz ile czasu to zajęło (liczba w strzałce). Pokazano również nazwę szpitala wraz z ilością wolnych łóżek przed oraz po ewentualnym przyjęciu pacjenta.

- Przykładowy format pliku wyjściowego

```
pacjent 0 :
(10,30) -10-> (5,10) szpital A 100/100 -> 100/100
(5,10) -5-> (10,20) szpital B 30/30 -> 30/30
(10,20) -20-> (30,40) szpital D 10/50 -> 11/50

pacjent 1 :
(12,50) -20-> (6,20) szpital E 120/120 -> 120/120
(6,20) -15-> (13,21) szpital F 25/30 -> 26/30
```

4 Testy

Program testowano w odrębych klasach testowych JUnit 4. Przyglądano się działaniu wyrażeń regularnych, poszczególnym metodom pomocniczym. Program testowano także jako całość podając różnepliki wejściowe (o różnych danych, rozmiarach, charakterystycznych cechach, szczególne przypadki)i sprawdzając pliki wyjściowe. Ponadto w tracie wykonywania programu na konsoli wyświetlane sąinformacje o zawartości struktur danych w poszczególnych iteracjach działania algorytmu. Stanowito proste logi programu, przydatne w procesie produkcji kodu oraz dla bardziej sprawnych przyszłych użytkowników. Wskazuje także na to czy program działa i ile iteracji potrzebuje na wykonanie zadania.

4.1 Test całościowy systemu

Ładowano pliki wejściowe podlegające testowaniu i sprawdzano w logach czy wykonywanie programu się kończy, w ilu iteracjach nastąpiło (efektywność).

5 Podsumowanie

Program został zaprojektowany tak, aby użytkownik mógł łatwo z niego korzystać podając plik mapy oraz plik pacjentów lub ręcznie podając pacjentów po kolei wraz z ich współrzędnymi. GUI pozwoli na wygodną i swobodną pracę z programem.