Specyfikacja implementacyjna
Projekt zespołowy

Mateusz Smoliński Łukasz Knigawka

23 grudnia 2018

Spis treści

1 Wstęp

Celem projektu jest napisanie programu prezentującego interaktywną mapę na podstawie danych z pliku wejściowego. Program otrzymuje punkty tworzące kontur mapy, punkty kluczowe, na podstawie których kontur zostaje podzielony na obszary, a także definicje obiektów oraz listę takowych obiektów, które mają być brane pod uwagę przy analizie wyżej wspomnianych obszarów.

Jeśli dane będą podane prawidłowo, program na ich podstawie generuje planszę, która dalej może być modyfikowana przez użytkownika. Może on dodawać i usuwać elementy konturu, zmieniając kształt mapy, a także dodawać i usuwać punkty kluczowe, wpływając tym samym na podział na obszary. Po kliknięciu na jeden z punktów kluczowych użytkownik otrzyma informacje o obiektach znajdujących się na wyznaczonym przez niego obszarze.

Projekt „LUPA" zostanie napisany w języku C# wersji 7.3 (.NET Framework 4.7.2), w środowisku Microsoft Visual Studio 15.9.4. Interfejs użytkownika zostanie utworzony dzięki platformie WPF. Implementacja oraz testowanie programu odbywać się będą na komputerach o następujących parametrach:

- 64-bitowy system operacyjny Windows 10 Home ver. 1803,
- procesor Intel Core i5-7200U,
- pamięć RAM 8,00 GB,
- karty graficzne Intel HD Graphics 620 + NVIDIA GeForce 940MX,

oraz:

- 64-bitowy system operacyjny Windows 10 Home ver. 1803,
- procesor Intel Core i7-6700HQ,
- pamięć RAM 16,00 GB,
- karty graficzne Intel HD Graphics 530 + NVIDIA GeForce 940MX,

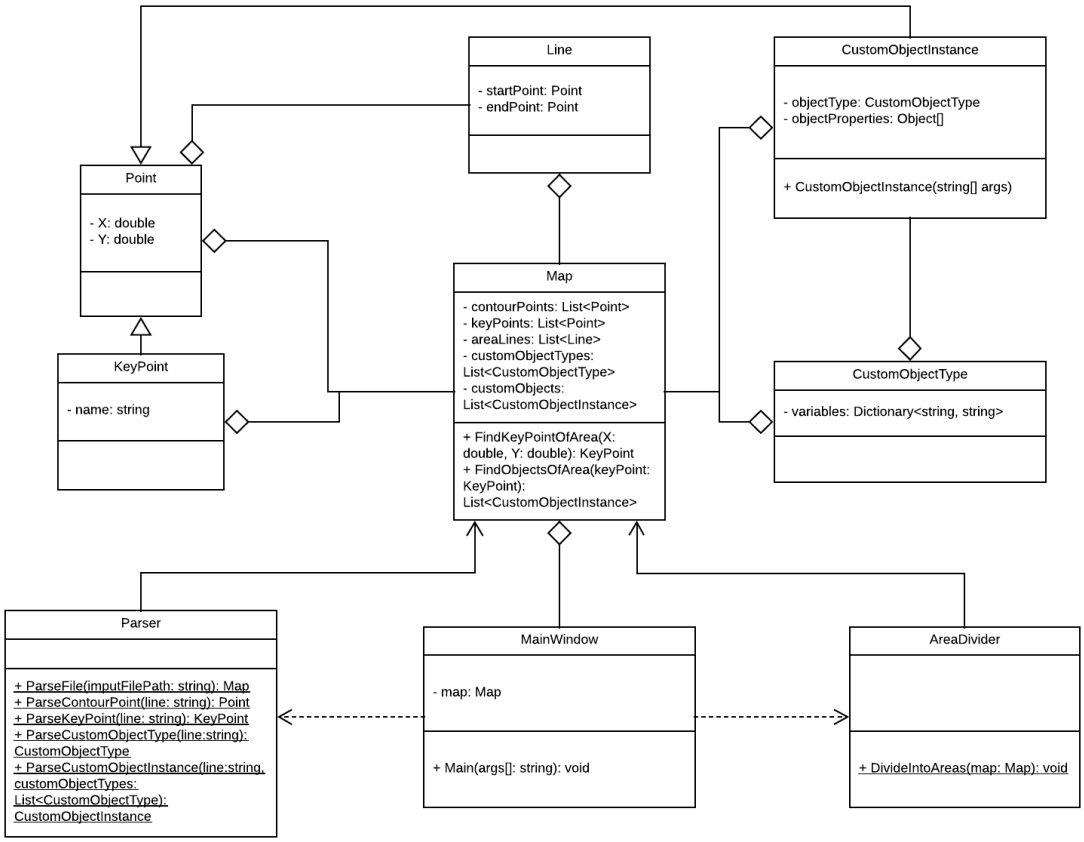
2 Diagram klas

Program składać się będzie z 9 klas. Zależności pomiędzy poszczególnymi klasami obrazuje Rysunek 1.

Na diagramie klas nie zostały uwzględnione:

- zależności pomiędzy klasami Parser, MainWindow, AreaDivider oraz klasami będącymi składowymi klasy Map, w celu poprawy czytelności,
- metody wygenerowane przez WPF, odpowiadające za rysowanie na planszy oraz reakcję na zachowanie użytkownika.

Nie występują na nim także metody dostępowe ze względu na zastosowanie automatycznie generowanych właściwości klas występujące w wybranym przez nas środowisku.



Rysunek 1: Diagram klas

3 Opis klas/metod

3.1 Klasa MainWindow

Jest to bazowa klasa tego programu, odpowiadająca przede wszystkim za interfejs graficzny. Składa się ona z wielu metod wygenerowanych przez WPF, odpowiadających za reakcję na działania użytkownika. Wśród najważniejszych logicznych implementacji, które muszą być zawarte w implementacjach tych metod, warto wspomnieć o:

- metodzie, która po kliknięciu w okolicy punktu kluczowego wypisze na panelach bocznych listy obiektów znajdujących się na obszarze wyznaczonym przez ten punkt kluczowy (pogrupowane i niepogrupowane według typu) oraz liczbę mieszkańców na tym obszarze,
- metodzie, która przy wybranej opcji punktu kluczowego po kliknięciu lewym przyciskiem myszy dodaje punkt kluczowy, a prawym przyciskiem usuwa najbliższy punkt kluczowy,
- analogicznej metodzie która dodaje i usuwa elementy konturu po wybraniu opcji konturu na pasku górnym.

Po wybraniu opcji usuwania/dodawania punktów na mapie następuje wywołanie funkcji *DivideIntoAreas* ponownie dzielącej mapę na obszary, z klasy *AreaDivider*.

3.2 Klasa 2

wq

3.2.1 Metoda 1

Mxq

3.3 Klasa 3

3.3.1 metoda

- `public CurrencyMatrix (int n)`

  standard listingu

3.4 Klasa 4

coś robi

3.5 Klasa 5

xxx

3.6 Klasa 6

W tej klasie coś

3.6.1 Metoda 1

3.6.2 Metoda 2

4 Działanie programu i zastosowanie algorytmu

Pełna sekwencja. Opis algorytmu może być problematyczny na tym etapie

5 Testy jednostkowe

Projekty testów zakładają użycie narzędzia ???. Poniżej znajduje się lista testów planowanych dla kluczowych metod programu, dla każdej z nich przewidziane są różne przypadki otrzymanych danych.

5.1 Testy klas 1 i 2

W tych dwóch nie trzeba wiele testować

5.2 Testy klasy 3

coś trzeba tu stestować

5.3 Testy klasy 4

Tu testujemy coś

5.4 Testy klasy 5

Ta klasa ma coś

5.5 Testy klasy 6

Ta klasa odpowiada za coś