**Parametry:**

* vector < vector < Stretch\* > > graph
  + dla każdej stacji, przechowuje wskaźniki na połączenia, które dotyczą danej stacji
* vector < Station\* > stations
  + przechowuje wskaźniki na każdą istniejącą stację
* vector < Stretch\* > connections
  + przechowuje wskaźniki na każde istniejące połączenie
* Area\* city
  + Wsakźnik na obiekt z klasy area
* int mStationsAmount
  + przechowuje ilość stacji
* int mStretchesAmount
  + przechowuje ilość połączeń

**Metody klasy:**

* **Konstruktory**
  + Graph()
    - Standardowy konstruktor
  + Graph(int)
    - Jako parameter przyjmowany jest rozmiar planszy (klasa Area), na której będziemy pracować
  + Graph(int,int)
    - Jako parametry metoda przyjmuje ilość stacji i połączeń, potem wywołuje metody addStation() i addStretch() odpowiednią ilość razy
* **Destruktor**
  + ~Graph()
    - Usuwa wszystkie wskaźniki na stacje/połączenia i zwalnia pamięć
* **Getter’y**
  + int getStationsAmount()
    - Zwraca ilość stacji
  + int getStretchesAmount()
    - Zwraca ilość połączeń
  + Area\* getArea()
    - Zwraca wskaźnik na obiekt klasy *Area*
  + vector < Station\* > getStations()
    - zwraca wektor wskaźników na wszystkie obiekty klasy *Station*
  + vector < Stretch\* > getStretches()
    - zwraca wektor wskaźników na wszystkie obiekty klasy *Stretch*
* **Metody dodające**
  + void addStation()
    - dodaje obiekt klasy *Station* do obiektu klasy *Graph*, wewnątrz metody zapytuje o parametry stacji
  + void addStation(string, int, int, int)
    - dodaje obiekt klasy *Station* o wprowadzonych parametrach
  + void addStretch()
    - dodaje obiekt klasy *Stretch* do obiektu klasy *Graph*, wewnątrz metody zapytuje o parametry połączenia
  + void addStretch(int, int, int)
    - dodaje obiekt klasy *Stretch* o wprowadzonych parametrach
* **Metody główne**
  + void toOnePoint(int)
    - symuluje jedno przejście ludności między stacjami w kierunku stacji określonej parametrem funkcji (pod warunkiem, że graf jest drzewem)
    - korzysta z funkcji DFS(int,int)
  + void action()
    - symuluje przejście między stacjami na dowolnym grafie
    - modyfikuje ilość ludzi na Stacjach
    - Korzysta z funkcji DFS2(bool\*, int\*, int, int, int) i Dijkstra(int)
  + void show()
    - wypisuje nazwę i ilość ludzi dla każdej stacji
  + void populationToStation()
    - wywołuje metodę klasy *Area,* która każdej stacji przypisuje ilość ludzi na podstawie mapy ludności (patrz klasa *Area*
* **Metody pomocnicze**
  + void DFS(int, int)
    - Wywoływana przez toOnePoint(int)
    - Przechodzi po grafie-drzewie i to ta metoda realnie zmienia ilości ludzi na stacjach
  + void DFS2(bool\*, int\*, int, int, int)
    - Wywoływana przez action()
    - Przechodzi po grafie I to ta metoda realnie zmienia ilości ludzi na stacjach
  + int\* Dijkstra(int)
    - Wywoływana przez action()
    - Dla wybranej stacji, zwraca tablicę odległości tej stacji od wszystkich pozostałych