# Sprawozdanie z projektu

Kacper Karaś, Jakub Niewadzi

5.06.2022

# Specyfikacja Funkcjonalna

## 1. Cel projektu

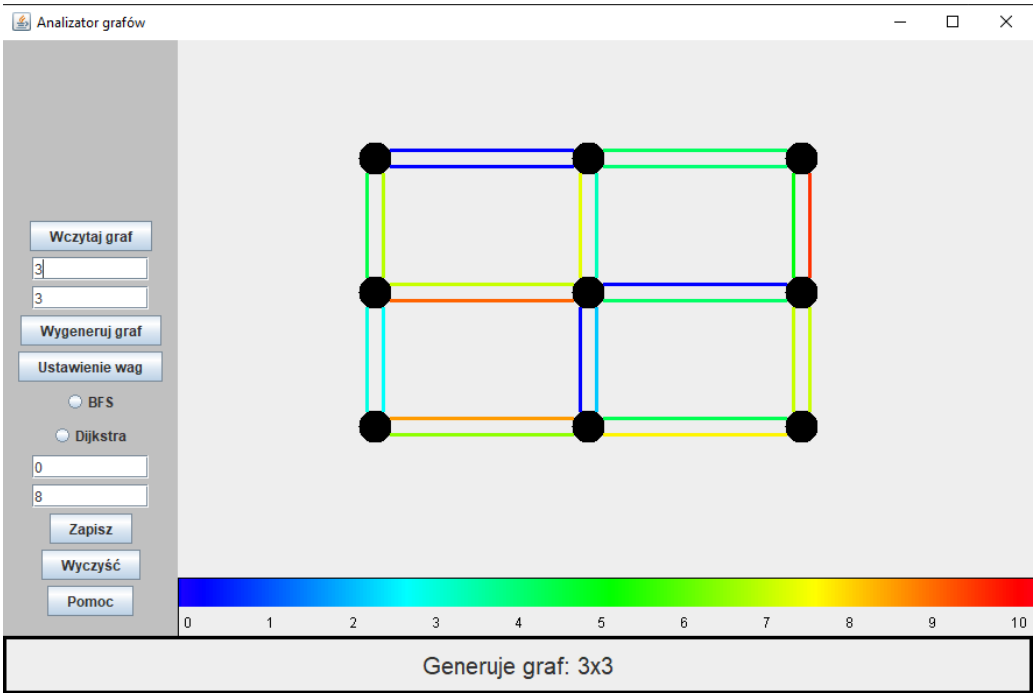
Program ma na celu generowanie, zapisywanie i czytanie grafów oraz powinien posiadać opcje sprawdzenia czy graf jest spójny za pomocą algorytmu BFS i wyznaczenia najkrótszej możliwej ścieżki pomiędzy wybraną parą węzłów, przy pomocy algorytmu Dijkstry.

## 2. Scenariusz działania programu

Program służy do badań grafu pod względem spójności wykorzystując algorytm BFS oraz znajduje w nim najkrótsze ścieżki za pomocą algorytmu Dijkstry. Program po uruchomieniu wyświetli nam okno interfejsu, gdzie po lewej stronie znajdują się przyciski z odpowiednimi funkcjami jakie może wykonać program. Wciśnięcie odpowiednich przycisków i podanie odpowiednich argumentów umożliwiających nam wykorzystanie programu w przewidzianym przez nas celu (więcej o formacie danych jak i argumentów będzie mowa w dalszej części specyfikacji). Jeżeli program nie znajdzie problemu przy wczytywaniu podanych przez nas argumentów, wyświetli graficzną interpretację grafu w centralnej części interfejsu. Wciśnięcie przycisku BFS lub Dijkstra rozpocznie analizę grafu odpowiednim algorytmem. Zostanie także zmieniona graficzna interpretacja grafu tak aby dobrze odwzorowywała wyniki działania algorytmów. Odpowiednie interpretacje graficzne zostaną szczegółowo opisane w dalszej części specyfikacji.

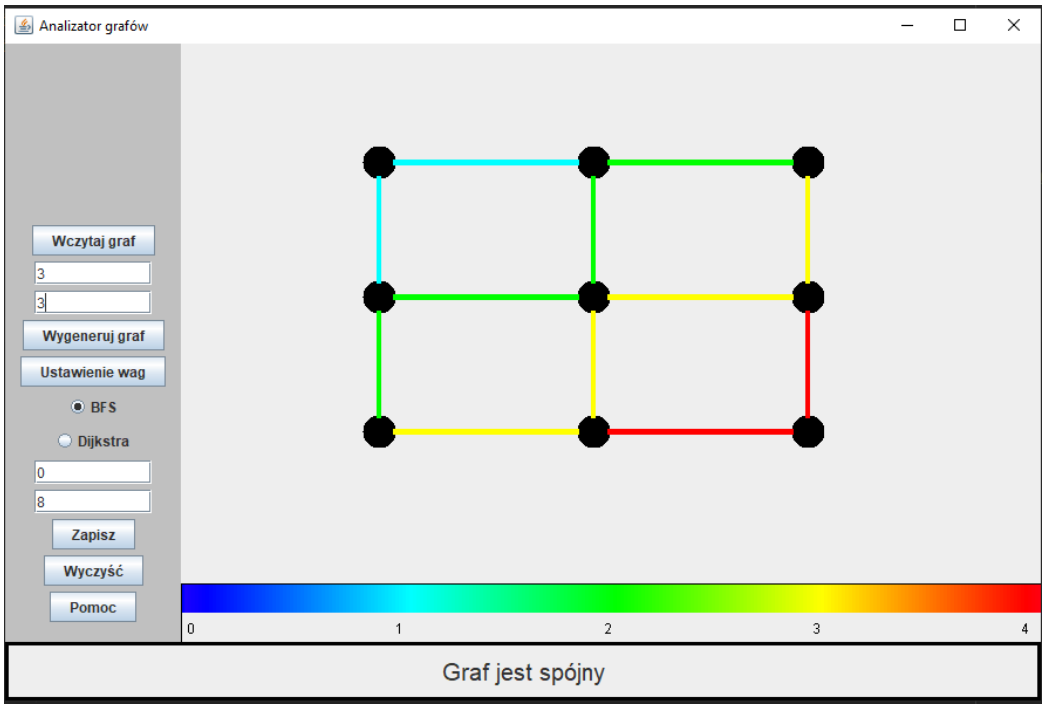
# 3. Interfejs programu

Interfejs programu po uruchomieniu programu i wgraniu pliku zawierającego graf:



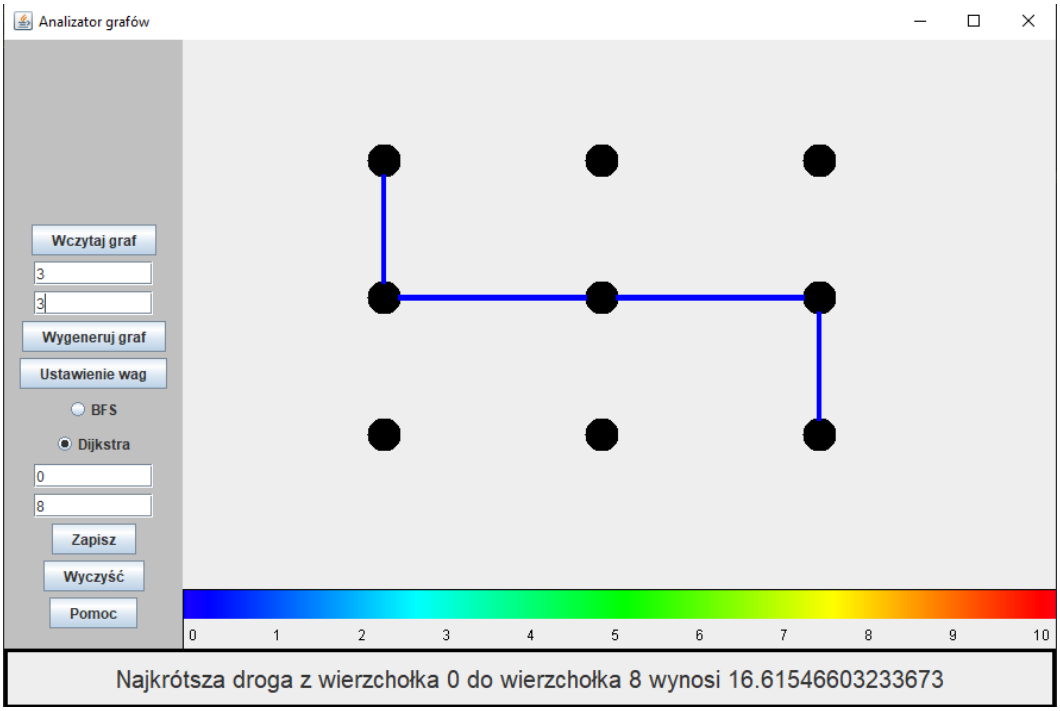

Kolory krawędzi oznaczają wagę jaka pomiędzy nimi występuje. Ścieżki w przestrzeni pionowej znajdujące się bardziej po lewej stronie to drogi z górnego wierzchołka do dolnego, prawe to drogi o dolnego do górnego. W przestrzeni horyzontalnej ścieżka znajdująca się na górze oznacza drogę między wierzchołkiem lewym i prawym, ścieżka dolna oznacza natomiast drogę z prawego wierzchołka do lewego. W przypadku gdy ścieżka do wierzchołka nie istnieje nie jest ona rysowana.

Interfejs programu po uruchomieniu programu i wgraniu pliku zawierającego graf i zanalizowania go algorytmem BFS:



Poszczególne kolory oznaczają odległość od wierzchołka 0 (górny prawy róg). Tak jak w przypadku rysowania całego grafu, gdy nie istnieje ścieżka do danego wierzchołka nie jest ona rysowana.

Interfejs programu po uruchomieniu programu i wgraniu pliku zawierającego graf i zanalizowania go algorytmem Djikstry:




Połączenia koloru niebieskiego oznaczają tu najkrótszą drogę pomiędzy dwoma wierzchołkami. Reszta połączeń nie jest rysowana. Aby zmienić wierzchołek docelowy należy kliknąć w pożądany wierzchołek lub wpisać jego numer w odpowiednie pole. Zmiana początkowego wierzchołka jest możliwa tylko przez wpisanie go w odpowiednie pole.

Opis odpowiednich przycisków Interfejsu:

- **Wczytaj graf** – po wciśnięciu przycisku wyświetli się nam okno wyboru pliku z komputera, z którego program będzie odczytywał graf
- **Wygeneruj graf** – po wciśnięciu przycisku program wygeneruje i narysuje graf o liczbie kolumn i wierszy podanych w pola powyżej przycisku
- BFS – po zaznaczeniu pola program rozpocznie analizę grafu algorytmem BFS, a następnie narysuje jego graficzną interpretację
- Djikstra - po zaznaczeniu pola program rozpocznie analizę grafu algorytmem Djikstra, a następnie narysuje jego graficzną interpretację. W polach poniżej przycisku można wpisać wierzchołek początkowy oraz końcowy dla algorytmu. Gdy nie jest podane inaczej początkowy wierzchołek jest zerowym a końcowy ostatnim wierzchołkiem grafu
- **Ustawienie wag** – po wciśnięciu przycisku wyświetli nam się okno, w którym należy podać wagi jakie chcemy, żeby miał generowany przez nas graf (program po uruchomieniu ma ustawione wagi 0 – 10)
- **Zapisz** – po wyciśnięciu przycisku wyświetli się nam okno wyboru pliku z komputera, w którym chcemy zapisać generowany graf
- **Wyczyść** - po wyciśnięciu przycisku wyświetla ponownie interfejs z podstawowym wyglądem grafu

- **Pomoc** - po wyciśnięciu przycisku wyświetla pomoc w jaki sposób prawidłowo korzystać z programu

# 4. Odpowiedni format argumentów

Podczas używania programu użytkownik będzie podawał pewną ilość argumentów, które powinny być w odpowiednim formacie:

- Program powinien mieć dostęp do pliku, z którego będzie czytał graf
- Rozmiary generowanego grafu powinny być liczbami naturalnymi dodatnimi
- Wagi podane przez użytkownika mogą być zmiennoprzecinkowe, ale muszą być nieujemne
- Gdy użytkownik poda plik do zapisu, który nie istnieje plik o takiej nazwie zostanie utworzony

# 5. Dane wejściowe

Program zakłada możliwość przekazania własnego pliku przechowującego informacje o ilości kolumn i wierszy w grafie oraz opisuje ilość krawędzi w każdym punkcie oraz ich wagi. Wagi krawędzi, gdy nie zostanie podany argument wejściowy mówiący inaczej, to liczby rzeczywiste z przedziału 0 – 10.

Plik ten powinien być sformatowany następująco:

- Dwoma pierwszymi liczbami powinno być liczba kolumn i wierszy grafu
- Następnie powinny pojawić się opisy wierzchołków, które występują w grafie (tu należy zaznaczyć, że numeracje zaczynamy od 0 i numerujemy w dół poszczególnych kolumn). Opisy wierzchołków powinny rozpoczynać się od pierwszego i kończyć na ostatnim, dlatego nie ma potrzeby zaznaczać jaki wierzchołek aktualnie opisujemy.
- Opis wierzchołka powinien składać się z co najwyżej 4 bloków zawierających numer sąsiadującego wierzchołka oraz wagę krawędzi między aktualnie wymienionym wierzchołkiem, a aktualnie opisywanym. Numer wierzchołka i waga powinny być od siebie oddzielone wyłącznie znakami spacji.

Przykładowy prawidłowo sformatowany plik:

```
3 3
    1 :1.232          3 :2.233
    0 :1.232          2 :0.485          4 :7.583
    1 :0.485          5 :6.593
    0 :2.233          4 :2.690          6 :9.515
    1 :7.583          3 :2.690          5 :10.000   7 :3.467
    2 :6.593          4 :10.000   8 :3.234
    3 :9.515          7 :7.383
    4 :3.467          6 :7.383          8 :8.487
    5 :3.234          7 :8.487
```

# 6. Teoria

Graf jest to uporządkowana para {**W, K**}, gdzie W to zbiór n węzłów odpowiednio ponumerowanych np. 0, 1, 2, …, n-1. K jest to zbiór krawędzi, czyli relacji między dwoma węzłami, które w naszym przypadku posiadają wagę (liczbę rzeczywistą).

Graf spójny - graf, dla którego spełniony jest warunek, że dla każdej pary węzłów istnieje ścieżka, która je łączy.

# 7. Komunikaty błędów

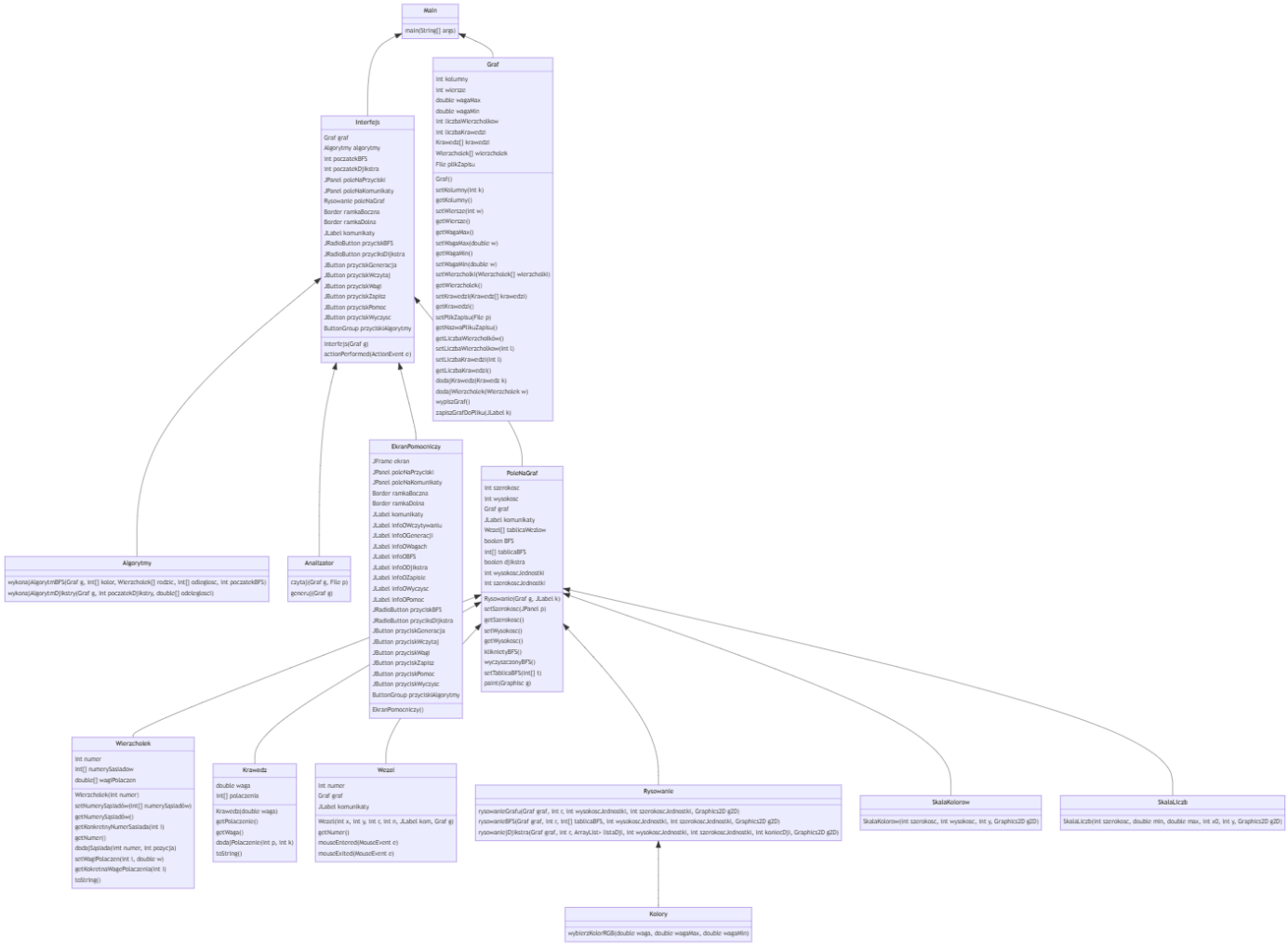
Program będzie kontynuował pracę po wpisaniu błędnych argumentów wtedy, kiedy jest to możliwe.

Oto lista komunikatów o błędach jakie mogą wystąpić podczas próby uruchomienia programu:

1. Nie wygenerowano / wczytano żadnego grafu aby rozpocząć analizę algorytmem.
2. Proszę podać liczby naturalną jako początek dla algorytmu Djikstra! Program automatycznie przypisuje początek jako wierzchołek 0.
3. Błędnie podano wymiary grafu!
4. Graf został wygenerowany ale jest za duży, żeby go narysować
5. Proszę wpisać w odpowiednie pola liczbę kolumn i wierszy! Muszą być to liczby naturalne > 0.
6. Błędny format pliku: nazwa_pliku
7. Proszę podać liczbę! (w przypadku niewpisania liczby przy ustalaniu wag)
8. Wagi powinny być liczbami dodatnimi!
9. Nie wygenerowano jeszcze grafu, który mozna było by zapisać!
10. Wystąpił błąd! Zapisywanie grafu zostało wstrzymane.
11. Proszę podać liczby naturalną dodatnią jako początek lub koniec dla algorytmu Djikstra! Koniec dla algorytmu nie może też przekraczać ilości wierzchołków w grafie.

# Specyfikacja implementacyjna programu.

## Klasy w programie:




Poniżej znajduje się opis klas jakie będą używane w projekcie wraz z krótkim jakie metody w sobie zawierają orz jakie argumenty zawiera (jeżeli jakieś zawiera):

- **Main** - zawiera inicjację klasy Graf oraz Interfejs. Posiada metodę main.

- **Graf** - jest to klasa zawierająca wszelkie informacje o grafie analizowanym przez program. Zawiera atrybuty przechowujące takie informacje jak: ilość kolumn, ilość wierszy, maksymalną wagę, minimalną wagę, liczbę wierzchołków, liczbę krawędzi, tablice klas Krawedzi i Wierzcholkow oraz plik, w którym będzie zapisywany graf. Tam, gdzie to potrzebne atrybuty mają napisane gettery i settery. Metody znajdujące się w klasie to: dodajKrawedz, dodajWierzcholek i wypiszGraf. Klasa zawiera również konstruktor, w którym ustawiona są podstawowe wartości wag.

- **Interfejs –** jest to najważniejsza klasa programu to dzięki niej wyświetlany jest interfejs i obiekty znajdujące się na nim. To również tu znajduje się ActionLisiner służący do wywoływania odpowiednich metod w programie w zależności od akcji użytkownika. Klasa ta zawiera inicjalizację wszystkich obiektów interfejsu oraz w konstruktorze rozmieszcza je odpowiednio w oknie głównym.

- **Krawendz** - jest to klasa zawierająca wszelkie informacje na temat poszczególnych krawędzi w grafie. Posiada argumenty takie jak: waga i tablice polaczenie (zawierającą dwa elementy pierwszy to numer początkowego wierzchołka krawędzi, drugi to numer końcowego wierzchołka. W konstruktorze klasy przypisuje się wagę danej krawędzi. Klasa posiada metodę doddajPolaczenie.

- **Wierzcholek** - jest to klasa zawierająca wszelkie informacje na temat poszczególnych wierzchołków w grafie. Posiada argumenty takie jak numer i tablice zawierającą numery sąsiadów (sąsiedzi przechowywanie są w następującej kolejności: pierwsze miejsce tablicy to górny sąsiad, drugie prawy sąsiad, trzecie dolny i czwarte miejsce lewy). W konstruktorze klasy przypisuje się numer danego wierzchołka oraz tablice sąsiadów wypełnia się wartościami -1. Klasa posiada matody: setNumerySąsiadów, getNumerySąsiadów, getNumer, dodajSąsiada.

- **Analizator –** jest to klasa zawierające metody związane z analizą grafu. Jedną z nich jest metoda czytaj odczytująca z podanego pliku dane na temat grafu i zapisujące je w odpowiednich argumentach klasy Graf. Druga metoda zajmuje się generacją grafu o określonej ilości kolumn i wierzchołków.

- **Algorytmy –** jest to klasa zawierająca metody analizujące graf zapisany w klasie Graf. Klasa posiada dwie metody: pierwszą analizującą graf za pomocą algorytmu BFS i drugą odpowiadającą za analizę grafu algorytmem Djikstry.

- **Rysowanie** – jest to klasa odpowiedzialna za rysowanie odpowiednich rodzajów grafów w centralnym polu w oknie głównym programu. Zawiera metody takie jak: rysowanieGeafu, rysowanieBFS, rysowanieDjikstra

- **Ekran pomocy** – jest to klasa odpowiadająca za wyświetlenie klonu interfejsu głównego, lecz z wyłączonymi przyciskami oraz z opisem co robi dany przycisk.

- **Kolory** – jest to klasa odpowiedzialna za wybór odpowiednych kolorów ze skali RGB w zależności od wagi krawędzi. Posiada statyczną metodę wybierzKolorRGB.

- **PoleNaGraf** – jest to klasa dziedzicząca klasę JPanel na której program rysuje graf oraz skalę wag krawędzi. Posiada konstruktor, w którym należy podać obiekt klasy GRAF, JLabel odpowiedzialny za komunikaty, oraz obiekt klasy Interfejs. To w tej klasie znajduje się metoda paint. Ponadto klasa posiada klasę wewnętrzną Wezel, która obsługuje MouseListenera i odpowiada za obsługę interakcji użytkownika z narysowanymi wierzchołakami.

- **SkalaKolorów** - jest to klasa odpowiedzialna za rysowania na dole PolaNaGeaf skali kolorów wykorzystywanej do pokazania jakie są wagi krawędzi narysowanego grafu. Posiada konstruktor, w którym należy podać oczekiwaną szerokość skali, jej wysokość, początkowy punkt y, z którego klasa rozpocznie rysowanie skali oraz obiekt klasy Graphics2D.

- **SkalaLiczb** – jest odpowiedzialna za dopisanie wartości liczbowych pod skalą kolorystyczną. Posiada konstruktor, w którym należy podać szerokość pola, na którym będą wypisywane liczby, minimalną liczbę, maksymalną liczbę, początkową zmienną x, zmienną y oraz obiekt klasy Graphics2D.

# Opis testów jednostkowych

## 1. AlgorytmyTest

Moduł “AlgorytmyTest” zawiera testy dla algorytmów BFS i Dijkstry. Przed wykonaniem testu dla poszczególnych algorytmów program generuje graf 25x25, który jest spójny.

Przy testowaniu algorytmu BFS sprawdzane są zwracane tablice – kolor, rodzic, odległość, czy mają odpowiednie dane dla grafu spójnego tj. Wszystkie elementy tablicy kolor są równe 2, jeden element tablicy rodzic jest równy null oraz czy jeden element tablicy odległości jest równy zero.

Do testowania algorytmu Dijkstry program ustawa wagi każdego połączenia na jeden i sprawdzane są wartości tabeli zwracanej przez algorytm Dijkstry - odległości.

## 2. AnalizatorTest

Moduł “AnalizatorTest” ma w sobie testy dla czytania i generacji grafów.

Test czytania polega na przeczytaniu pliku “test1” znajdującego się w repozytorium i porównaniu wyników czytania z zawartościami tablic poprawneWagi i poprawniSasiedzi, w których zapisane są poprawne tablice, które powinien posiadać w sobie graf. Sprawdzane są wagi poszczególnych krawędzi w odpowiedniej kolejności oraz pierwszy sąsiad dla każdego wierzchołka.

Test generacji generuje graf 100x100, który posiada zakres wag z przedziału od 0.0 do 10.0, sprawdza czy w generacji wystąpiła waga większa od 9.8 i mniejsza od 0.2. Sprawdzane są tablice krawędzi i wierzchołki w grafie, czy nie zawierają w sobie wartości null.