Sprawozdanie końcowe z projektu w języku Java

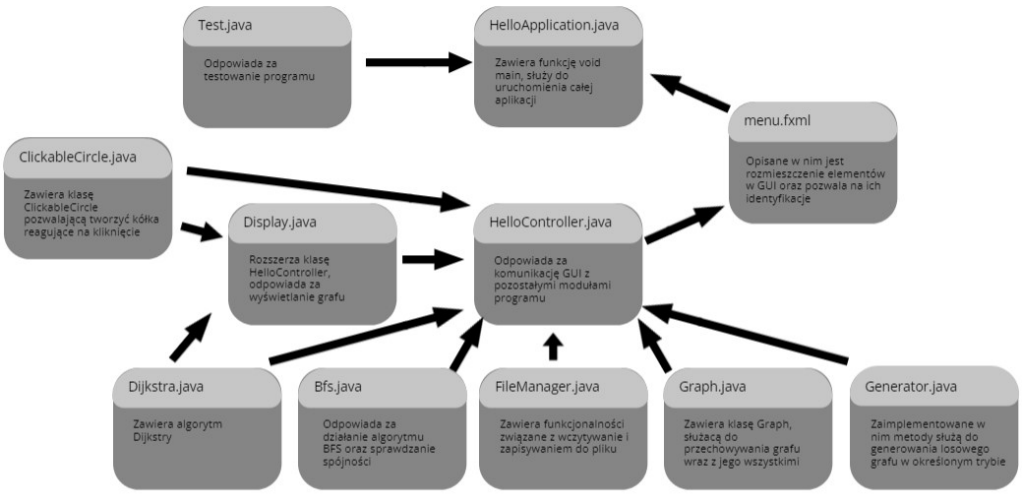
Kamil Fryszkowski, Oskar Biwejnis

8 czerwca 2022

1 Cel projektu

Celem projektu było napisanie w dwuosobowym zespole programu w języku Java posiadającego graficzny interfejs użytkownika, zbudowany przy użyciu technologii JavaFX, pozwalającego: generować na podstawie danych parametrów albo czytać z pliku graf i wyświetlać jego reprezentacje graficzną, także sprawdzać jego spójność algorytmem BFS oraz obliczać najkrótsze ścieżki pomiędzy danymi punktami za pomocą algorytmu Dijkstry. Miało to nas nauczyć podstaw projektowania GUI, tego jak sprawić aby komunikacja programu z użytkownikiem była efektowna oraz współpracy w pisaniu kodu na zdalnym repozytorium w systemie kontroli wersji GIT.

2 Struktura programu



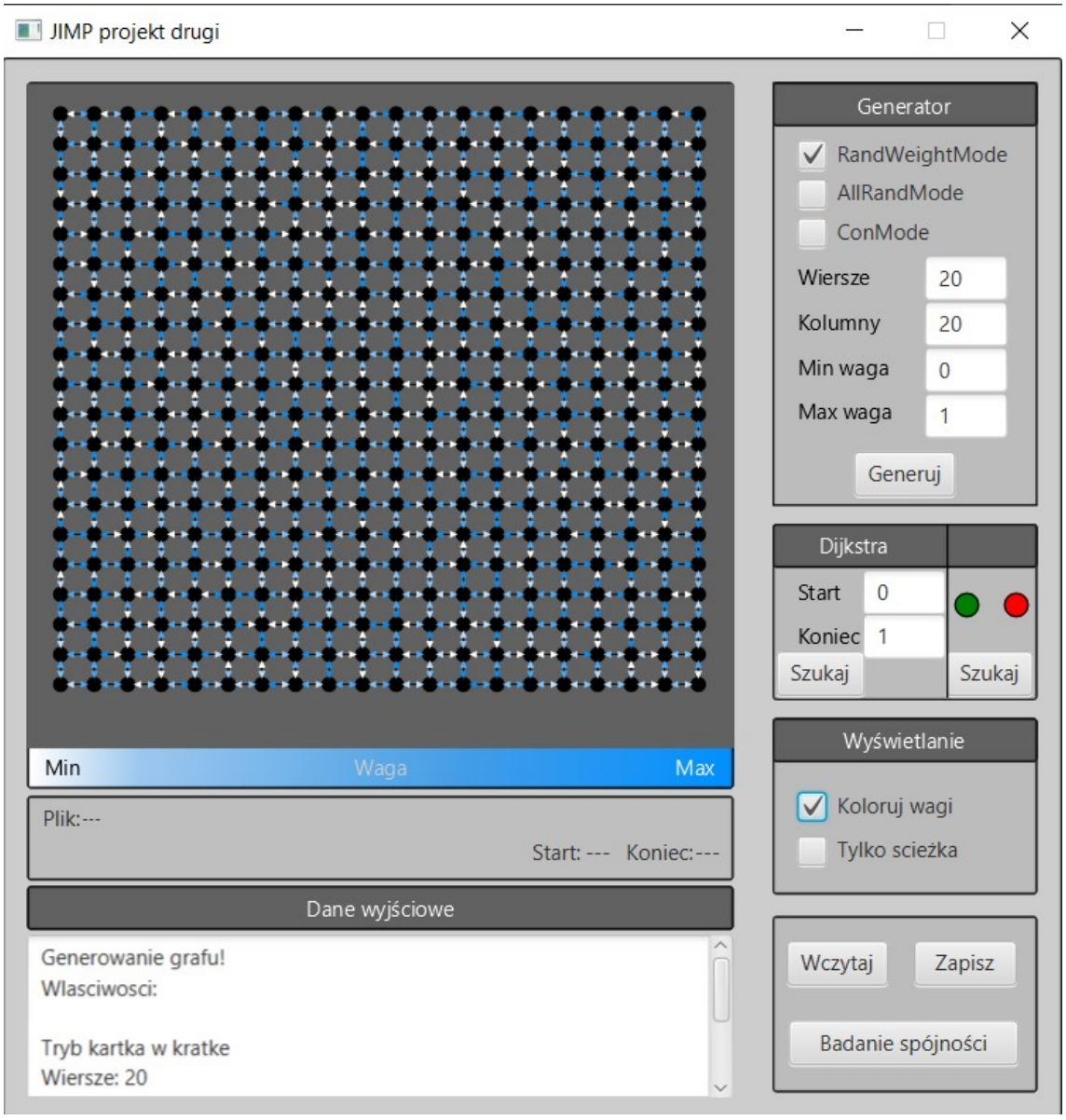
Rysunek 1: Graficzna reprezentacja struktury programu (opracowanie własne)

- HelloApplication.java odpowiada za odpowiednie uruchomienie programu, uruchamianie testów i wczytanie sceny
- Test.java zawiera testy metod generujących graf, zapisujących i wczytujących plik, BFSa oraz Dijkstry.
- menu.fxml to plik zawierający w sobie wygląd GUI
- HelloController.java to plik który steruje całym GUI
- Display.java to plik, który zawiera metody wykorzystywane przy sterowaniu GUI. Powstał po to, aby HelloController.java był krótszy i czytelniejszy.
- ClickableCircle.java dziedziczy po standardowej klasie Circle oraz zawiera on w sobie definicje wyświetlanych w GUI wierzchołków

- Dijkstra.java zawiera metody pozwalające na szukanie najkrótszych ścieżek
- BFS.java zawiera metody pozwalające na sprawdzenie czy graf jest spójny
- FileManager.java zawiera metody które wczytują lub zapisują graf do/z pliku
- Graph.java zawiera definicje przechowywania grafu
- Generator.java zawiera metody generujące graf

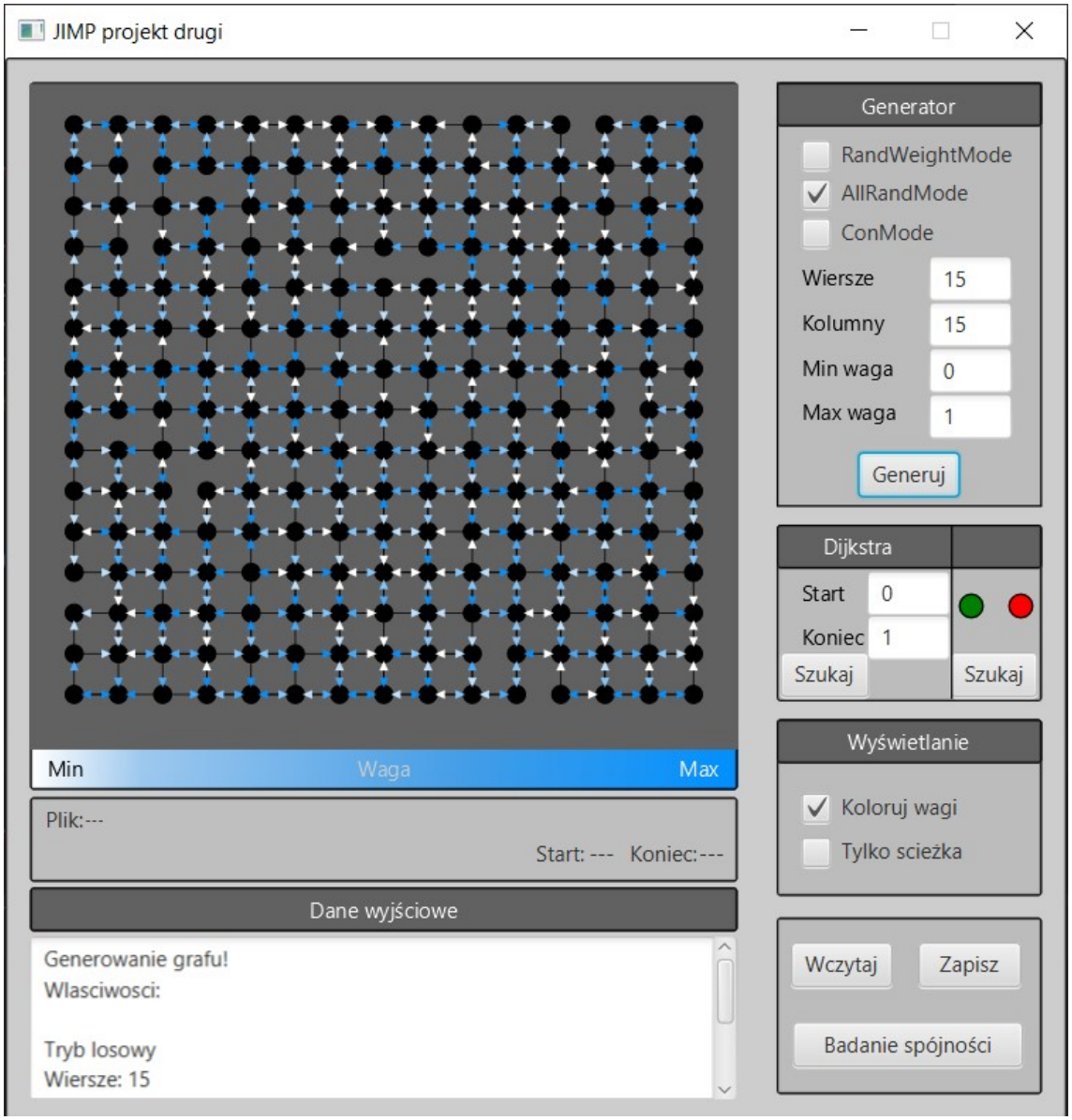
3 Przykładowe wywołania programu

3.1 randWeightMode



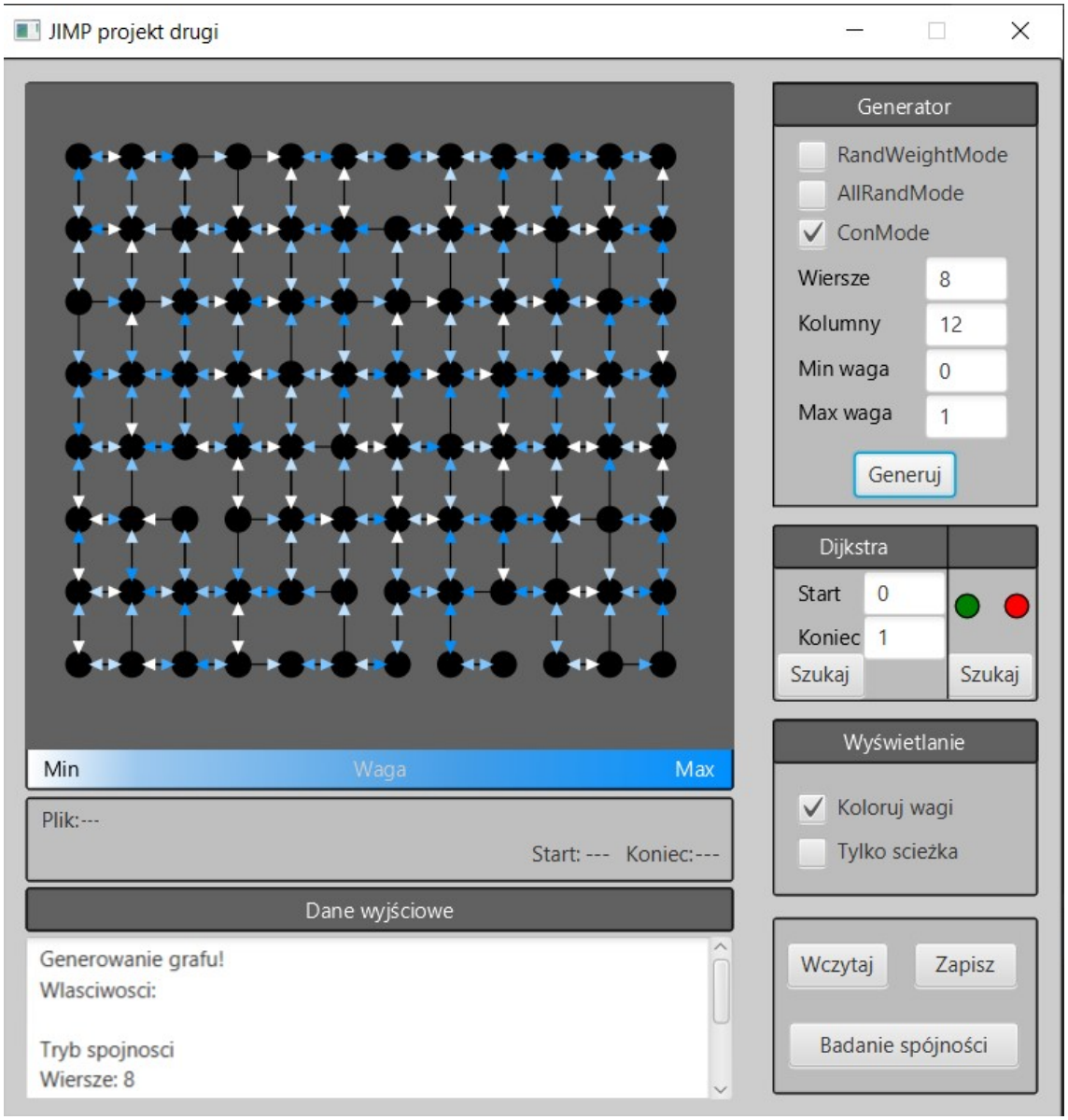
Rysunek 2: Graf wygenerowany w trybie losowych wag

3.2 allRandMode



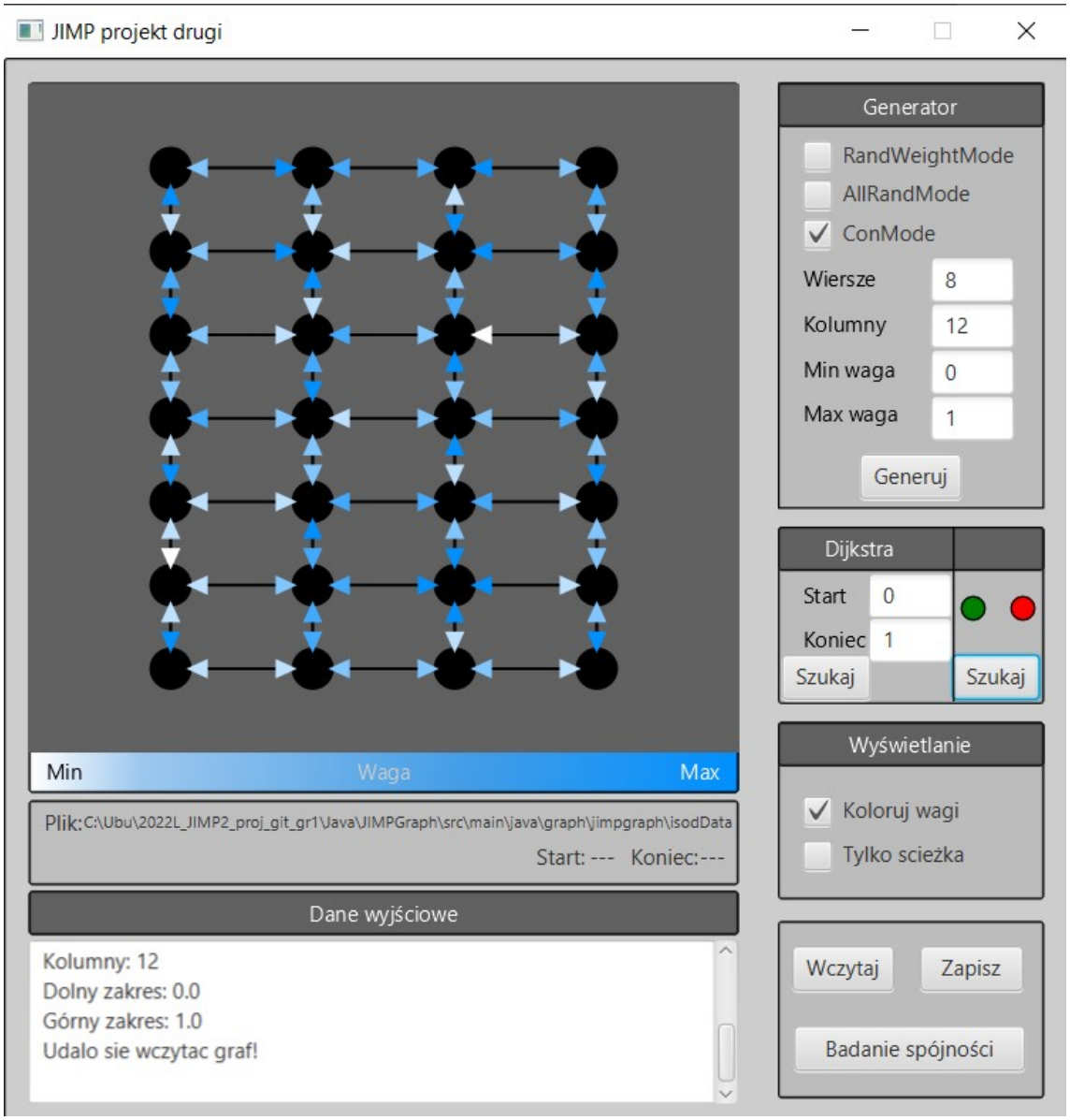
Rysunek 3: Graf wygenerowany w trybie całkowicie losowym

3.3 conMode



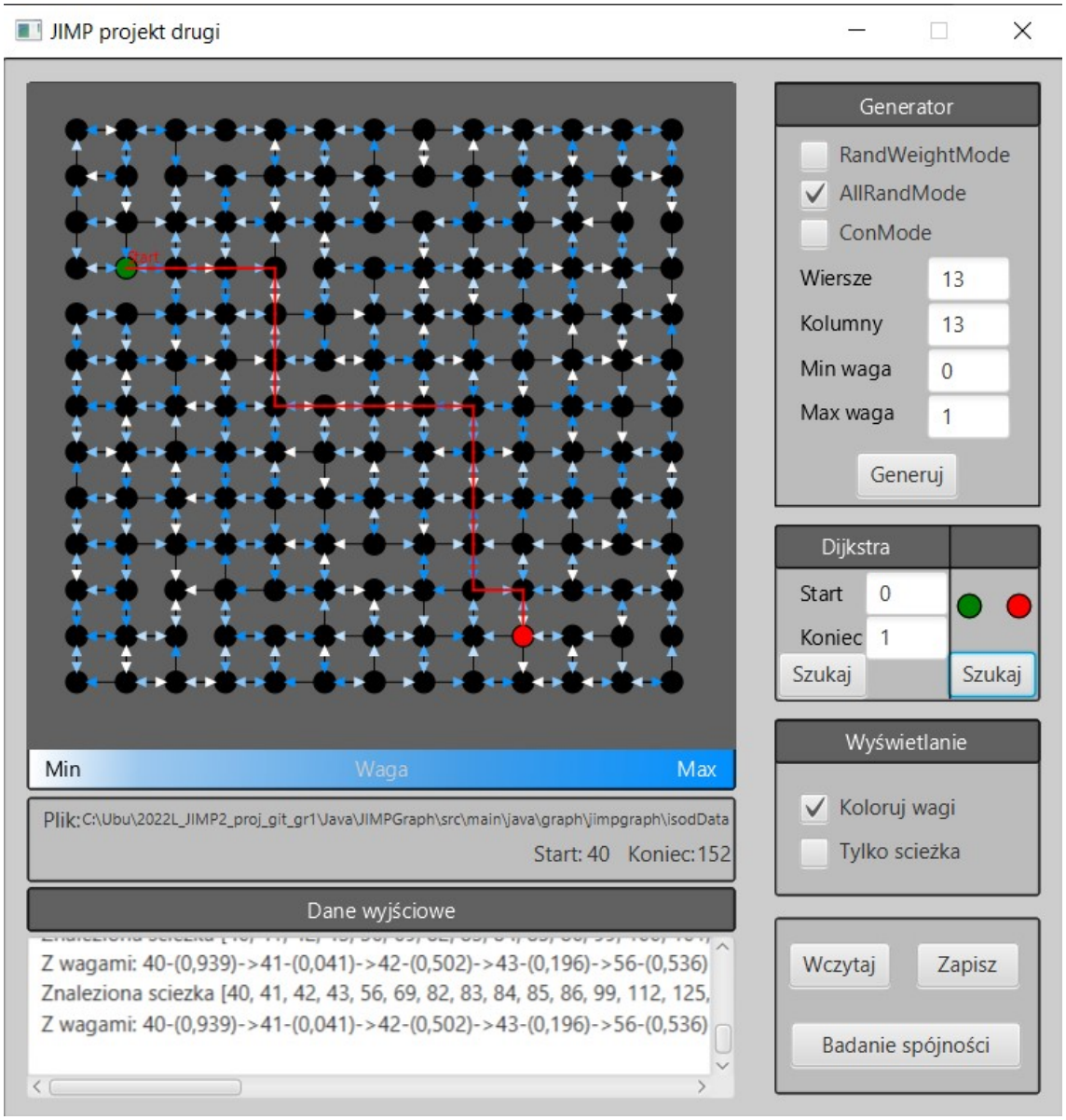
Rysunek 4: Graf wygenerowany w trybie spójności

3.4 loadMode

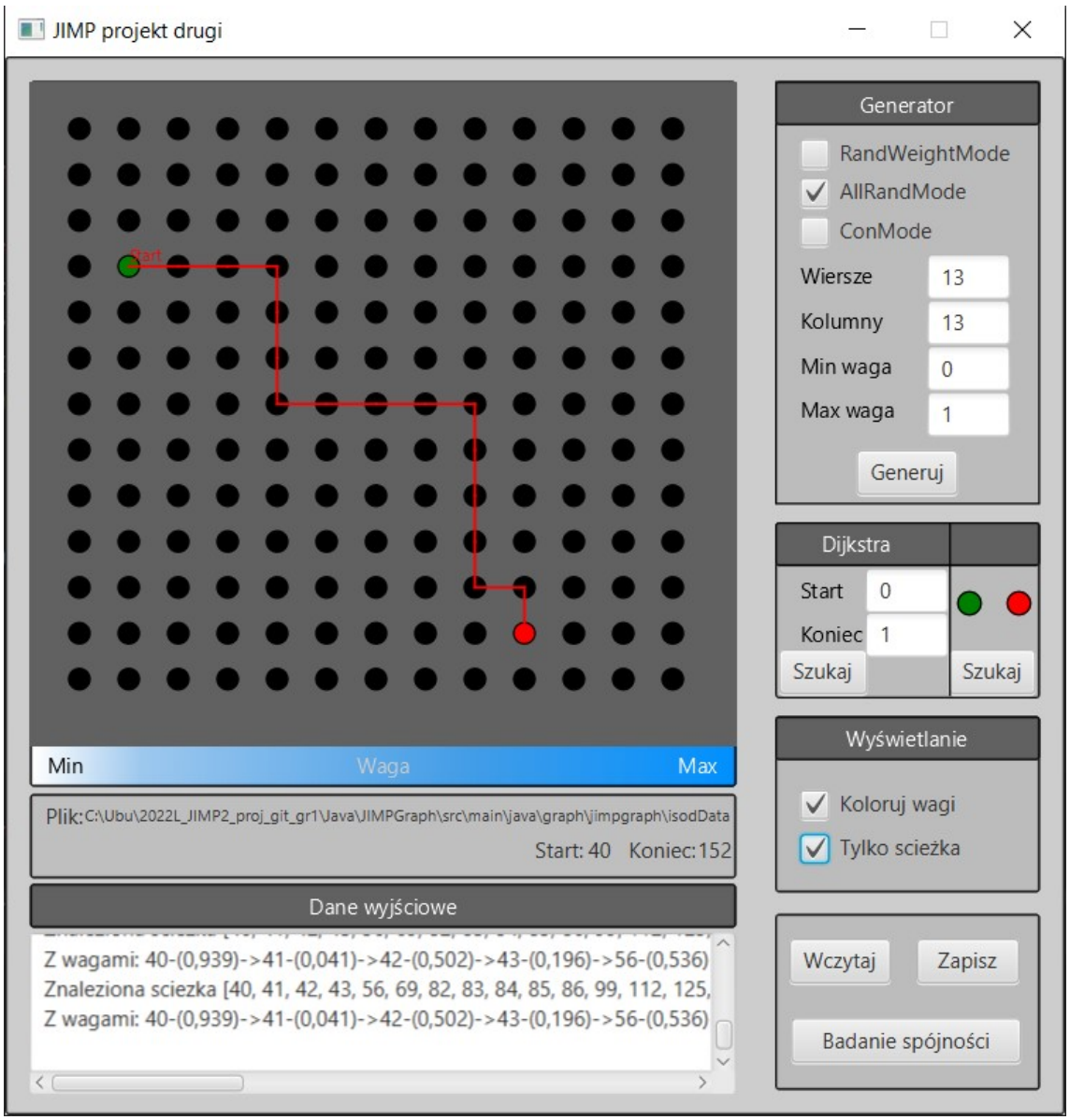


Rysunek 5: Wczytany graf z plików z Isod

3.5 Dijkstra



Rysunek 6: Wyświetlanie wyniku algorytmu Dijsktry



Rysunek 7: Włączona opcja pokazująca jedynie ścieżkę

4 Zmiany w ostatecznej wersji względem pierwotnej

Zmianie uległ schemat klas oraz zawartości poszczególnych klas

- `app.java` zostało zmienione w trzy osobne klasy, to znaczy `HelloAplication.java Display.java HelloController.java`.
- `Gui.fxml` zmieniło nazwę na `menu.fxml`.
- `Generator.java` nie zawiera tak jak w specyfikacji struktury grafu. Do tego celu została stworzona osobna klasa `Graph.java`.
- `File.java` zmieniło nazwę na `FileManager.java` ponieważ używaliśmy standardowej klasy `File` by otworzyć konkretny plik.
- Powstała nowa klasa `ClickableCircle.java`, która dziedziczy po standardowej klasie `Cricle`.

5 Podsumowanie i wnioski

Projekt można uznać za zakończony, a jego finalną wersję ocenić pozytywnie. Nauka, w miarę systematyczna praca, wspólna komunikacja oraz wyciąganie odpowiednich wniosków pozwoliły stworzyć program, który niemal całkowicie spełnia założenia jakie sobie postawiliśmy.
W programie pojawiło się kilka mniej udanych rozwiązań, przede wszystkim testy nie zostały napisane przy pomocy jUnit, a pokazywanie grafu skalowało się do określonej wielkości okna, przez co duże grafy były tak małe że aż niewidoczne. Cała reszta to kosmetyczne kwestie nie wpływające na samą funkcjonalność, czyli sporadyczna, miejscowa niedokładność lub niespójność formatowania kodu.
Gdybyśmy mieli pisać ten projekt jeszcze raz, to postaralibyśmy się stworzyć widok grafu taki, żeby nawet przy bardzo dużych rozmiarach był widoczny lub możliwy do przybliżenia. Postaralibyśmy się również, aby nie zajmować niepotrzebnie pamięci dla wierzchołków, które nie istnieją, to znaczy powyżej pierwszego wiersza i tym podobnych. Spróbowalibyśmy również napisać cały program w metodyce TDD, koniecznie z wykorzystaniem jUnit (powodem dlaczego od razu nie korzystaliśmy z tej metodyki jest fakt, że to była nasza pierwsza styczność z tworzeniem aplikacji z interfejsem graficznym). Porównując projekt do poprzedniego, to znaczy aplikacji konsolowej w języku C zauważiliśmy, że jest dużo mniej zmian w porównaniu do specyfikacji. Uważamy, że wynika to z prostego faktu, że de facto tworzyliśmy bardzo podobny program wykorzystując jedynie inny język programowania, jakim jest JAVA, a sam trzon aplikacji opiera się na tych samych zależnościach.

6 Przebieg współpracy

Współpracę można ocenić pozytywnie. Dzięki jasno ustalonym wymaganiom, które sobie określiliśmy w specyfikacjach, praca nad programem obyła się bez większych problemów. Komunikacja przebiegała sprawnie od samego początku. Równomiernie podzieliliśmy zadania w zespole i na bieżąco informowaliśmy siebie nawzajem o postępach.
Pracowaliśmy z pomocą systemu kontroli wersji GIT wykorzystując repozytorium znajdujące się na Projektorze. Każdy z nas miał swoją własna gałąź, na której pracował, a po dokonaniu weryfikacji kodu przez drugą osobę, zmiany były dodawane do gałęzi głównej.