Dokumentacja Końcowa
Projektu Labirynt JAVA

Polina Nesterova i Martyna Kochalska

03 czerwca 2024

Streszczenie

Ten dokument jest sprawozdaniem z postępów prac nad projektem labiryntu w języku JAVA. Zawiera informacje o głównym celu projektu, szczegółowy opis struktury katalogów, diagram modułów oraz prezentację wyników uzyskanych podczas uruchamiania programu. Dodatkowo, dokonano podsumowania całego projektu oraz analizy współpracy.

Contents

1 Cel projektu

Celem projektu jest stworzenie programu w języku Java z graficznym interfejsem użytkownika opartym na technologii SWING. Program ma umożliwić wczytywanie labiryntów z plików tekstowych i binarnych, znalezienie najkrótszej ścieżki od punktu wejścia do wyjścia w labiryncie oraz zapisanie labiryntu wraz z rozwiązaniem.

W interfejsie użytkownika znajduje się panel narzędziowy umożliwiający manipulację labiryntem, w tym zmianę punktu wejścia i wyjścia, oraz prezentacja wczytanego labiryntu i znalezionej ścieżki. Ponadto program powinien być w stanie zapisać labirynt z rozwiązaniem i bez rozwiązania w formacie PNG.

Przykładowy format labiryntu 5 x 4 podany w wersji tekstowej:

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
P X       X             X       X             X
X X XXX X XXX XXX X XXX X X X XXX X X XXX
X   X X   X X     X     X   X     X X   X
X X X X XXX X XXXXXXXXXXXXXXX XXXXXXXXX X
X X   X     X                           X X
X XXX X XXX X XXXXXXX X XXXXX XXXXXXX X X
X X             X   X X   X       X   X X
X X XXXXX X XXX XXX X XXXXX XXX X XXX X X
X         X                       X   X   K
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Wyjście z programu

Program powinien generować listę wykonanych kroków, która pokaże drogę przez labirynt. Przykładowa lista kroków może wyglądać następująco:

- FORWARD 1
- TURNLEFT
- FORWARD 4
- TURNRIGHT
- FORWARD 3

Obsługa wczytywania i zapisywania labiryntu w postaci pliku binarnego

Program musi umożliwiać wczytywanie labiryntu z pliku binarnego oraz zapisywanie go w tej formie. To obejmuje prawidłowe przetwarzanie danych binarnych zgodnie z określoną specyfikacją.

Pobieranie plików PNG

Program powinien umożliwiać użytkownikowi zapisywanie labiryntu wraz z jego rozwiązaniem w formacie PNG. Ten format pliku powinien zawierać graficzną reprezentację labiryntu wraz z zaznaczoną znalezioną ścieżką.

Maven

Projekt powinien być zarządzany za pomocą narzędzia Apache Maven, co ułatwi zarządzanie zależnościami, kompilację, testowanie i budowę programu. Pomocne byłoby też utrzymanie jednolitej struktury projektu i konfiguracji. Opracowanie programu spełniającego powyższe cele będzie kluczowe dla sukcesu projektu. Wszelkie dodatkowe funkcje lub ulepszenia będą mile widziane, ale należy najpierw zapewnić, że program spełnia podstawowe wymagania.

2 Opis plików wykorzystywanych przez program

- `Pliki Wejściowe` – program korzysta z plików zawierających informacje o labiryncie. Możliwe są dwa rodzaje plików wejściowych: tekstowy (z rozszerzeniem '.txt') oraz binarny (.bin).

  Plik tekstowy zapisany powinien być za pomocą znaków 'X', ' ', 'P' i 'K', gdzie znaki te oznaczają kolejno ścianę, przejście, początek i koniec labiryntu.

  Plik binarny składa się z 4 głównych sekcji:

  1. Nagłówka pliku
  2. Sekcji kodująca zawierająca powtarzające się słowa kodowe
  3. Nagłówka sekcji rozwiązania
  4. Sekcji rozwiązania zawierające powtarzające się kroki które należy wykonać aby wyjść z labiryntu

  Dokładny opis pliku binarnego jest możliwy do pobrania tutaj: https://isod.ee.pw.edu.pl/

- `Pliki Wyjściowe` – program generuje plik wyjściowy, w którym zapisywane są wyniki działania programu. Taki plik jest opcjonalny i jego nazwa może być podana przez użytkownika jako argument wywołania.

3 Struktura programu

3.1 Struktura folderów

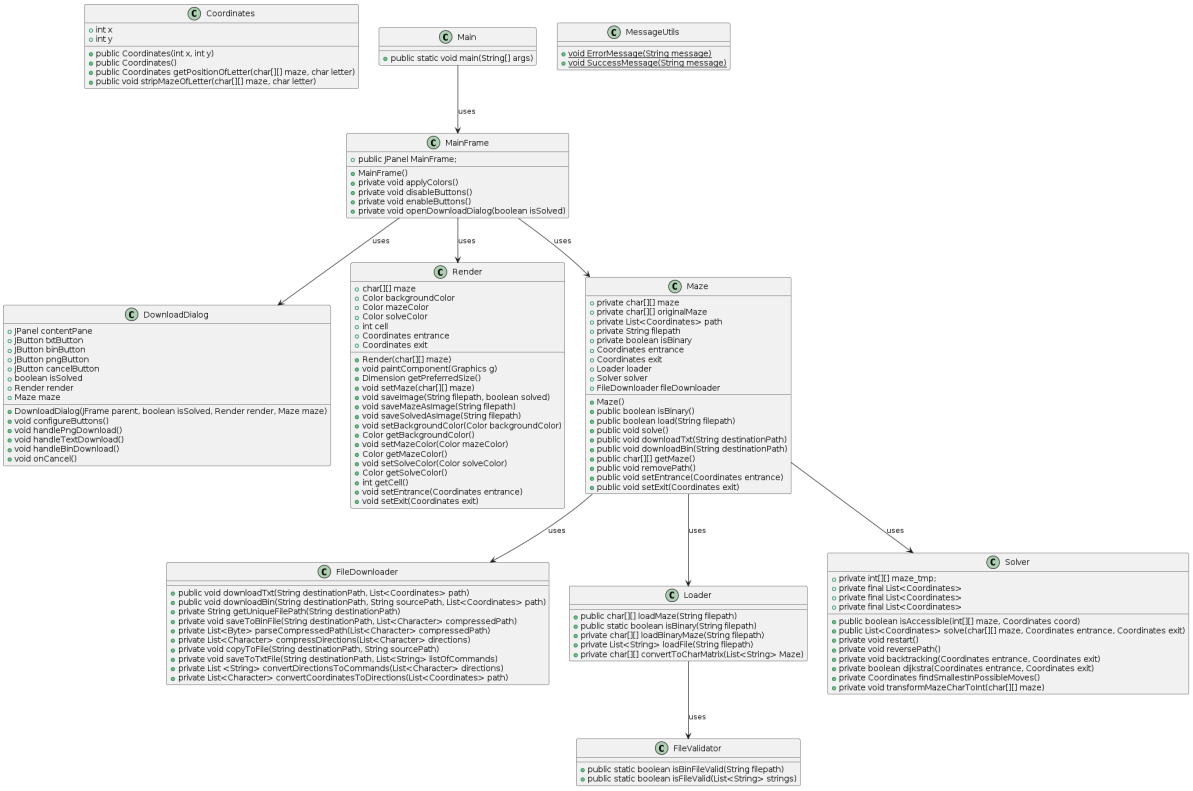
W projekcie wykorzystywana jest domyślna struktura katalogów meaven, dodatkowo zostały załączone foldery

- documentation

- default_maps

przechowywujące przykładowe pliki wejściowe oraz wymaganą dokumentację.

3.2 Diagram modułów



Program składa się kolejno z modułów:

1. `MainFrame` - Obsługa GUI, przycisków.
2. `Render` - Generowanie wizualnej reprezentacji labiryntu, ścieżki, wyjść i wejść.
3. `Maze` - Przechowywanie informacji dotyczących labiryntu, pośrednik w komunikacji pomiędzy MainFramem i Loaderem, Solverem.
4. `Loader` - Załadowywanie pliku tekstowego i binarnego z pliku.
5. `FileValidator` - Sprawdzenie poprawności załadowanych plików binarnych i tekstowych.

6. `Solver` - Znalezienie najkrótszej ścieżki w labiryncie przy pomocy implementacji algorytmu Djikstra.
7. `Downloader` - Zapis ścieżki do pliku, przepisanie labiryntu na wymaganą formę dla pliku binarnego.
8. `MessageUtils, Coordinates` - Moduły pomocnicze dostępne dla każdego modułu, MessageUtils umożliwia wyświetlenie wiadomości, Coordinates służy jako struct trzymający w sobie koordynaty x i y.

4 Opis działania modułów

- `Moduł 1 - MainFrame`

  Główne funkcjonalności:

  1. Wyświetlenie panelu z interfejsem
  2. Oczekiwanie na input w postaci naciśnięcia przycisku i komunikacja z pozostałymi klasami.
  3. Współpracuje z klasami: Maze, Render, MessageUtils

- `Moduł 2 - Render`

  Główne funkcjonalności:

  1. Wyświetlanie labiryntu, ścieżki
  2. Aktualizacja kolorów wyświetlanego labiryntu
  3. Zapisywanie wygenerowanego labiryntu jako png.
  4. Współpracuje z klasami: MainFrame(wywoływany), Coordinates

  Metody publiczne:

  – public void saveMazeAsImage(String filepath) throws IOException - umożliwia pobranie obrazu mapy labiryntu do lokalizacji podanej w argumencie.

  – public void saveSolvedAsImage(String filepath) throws IOException - umożliwia pobranie obrazu mapy labiryntu razem z rozwiązaniem do lokalizacji podanej w argumencie.

  – public void setExit/Entance(Coordinates exit/entrance) - setter do wejścia wyjścia w klasie.

- `Moduł 3 - Maze`

  Główne funkcjonalności:

  1. Przechowywuje dane dotyczące labiryntu: ścieżkę, wejście, wyjście, mapę.

2. Pośredniczy w komunikacji między klasami
3. Współpracuje z klasami: MainFrame(wywoływany), Solver, Loader, FileDownloader, Coordinates, MessageUtils

Metody publiczne:

- boolean load(String filepath) – Wywoływane przez MainFrame, wywołuje metodę loadMaze z Loadera, przypisuje zmiennym maze, entrance, exit odpowiednie wartości, przy błędzie zwraca false, przy poprawnym załadowaniu - true.
- void solve(String filepath) - Wywołuje Solver.solve, przypisuje wartości ścieżce path.
- void download(String filepath) - wywołuje odpowiednio funkcję SavePathToBin() lub SavePath w zależności od pliku wejściowego.
- public void removePath() - usuwa ścieżkę po wywołaniu.
- public void restorePath() - przywraca ścieżkę.
- Dodatkowo: gettery i settery do: Maze, path, entrance, exit.

- `Moduł 4 - Loader`

  Główne funkcjonalności:

  1. Rozpoznanie rodzaju pliku wejściowego
  2. Wywołanie klasy FileValidator
  3. Wczytanie pliku tekstowego i zwrócenie go w odpowiedniej formie.
  4. Wczytanie pliku binarnego i zwrócenie go w odpowiedniej formie.
  5. Współpracuje z klasami: Maze(wywoływany), FileValidator, Coordinates.

  Metody publiczne:

  - public char[][] loadMaze(String filepath) - metoda otrzymująca ścieżkę do pliku wejściowego, sprawdzająca typ pliku, wywołująca klasę sprawdzającą jego poprawność, wywoływująca funkcje załadowywujące i korygujące formę pliku, zwracająca mazeMatrix[][].
  - public static boolean isBinary(String filepath) - metoda sprawdzająca typ pliku (binarny/tekstowy)

- `Moduł 5 - FileValidator`

  Główne funkcjonalności:

1. Odczyt danych zapisanych w pliku wejściowym binarnym i tekstowym, sprawdzenie ich poprawności.
2. W razie błędu wywołuje informację przy pomocy MessageUtils.
3. Współpracuje z klasami: Loader(wywoływany), MessageUtils, Coordinates.

Metody publiczne:

- public static void ErrorMessage(String message) - metoda wywołująca popup z ikoną błędu wyświetlający wiadomość podaną w argumencie.

- public static void SuccessMessage(String message) - metoda wywołująca popup z ikoną informacji wyświetlający wiadomość podaną w argumencie.

- `Moduł 6 - Solver`

  Główne funkcjonalności:

  1. Przeskanowanie labiryntu w poszukiwaniu ścieżek
  2. Znalezienie najkrótszej ścieżki i zwrócenie jej.
  3. Współpracuje z klasami: Maze(wywoływany), MessageUtils, Coordinates.

  Metody publiczne:

  - public List<Coordinates> solve(char[][] maze, Coordinates entrance, Coordinates exit) - wywołuje metody rozwiązujące labirynt, zwraca ścieżkę.

- `Moduł 7 - FileDownloader`

  Główne funkcjonalności:

  1. Konwertuje ścieżkę na format pliku binarnego (literka N/S/W/E + iloś kroków zapisana bitowo w 1 bajcie) i zapisuje ją do niego.
  2. Konwertuje ścieżkę na format TURN RIGHT/LEFT, FOWARD [liczba] i zapisuje wynik do pliku.
  3. Współpracuje z klasami: Maze(wywoływany), Coordinates, MessageUtils.

  Metody publiczne:

  - public void savePathtoTxt(List<Coordinates> Path, String filename) - wywołuje funkcję konwertującą na odpowiedni format i zapisuje do pliku o ścieżce filename.

  - public void savePathtoBin(List<Coordinates> Pathm) - wywołuje funkcję konwertującą na format i zapisuje do pliku wejściowego.

- `Moduł 8 - Coordinates`

  Główne funkcjonalności:

  1. Klasa przechowywująca koordynaty x, y

- `Moduł 9 - MessageUtils`

  Główne funkcjonalności:

  1. Umożliwia innym modułom wyświetlenie informacji o stanie operacji po akcji na interfejsie graficzym.

  Metody publiczne:

  - public void savePathtoTxt(List<Coordinates> Path, String filename) - wywołuje funkcję konwertującą na odpowiedni format i zapisuje do pliku o ścieżce filename.

  - public void savePathtoBin(List<Coordinates> Pathm) - wywołuje funkcję konwertującą na format i zapisuje do pliku wejściowego.

5 Kompilacja programu

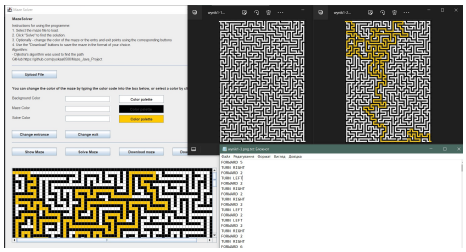
Aby uruchomić aplikację, użytkownik musi jedynie wejść do podfolderu „out/artifacts/Maze_JAVA_jar" w katalogu aplikacji i uruchomić plik Maze_JAVA.jar, dwukrotnie klikając go.

6 Przykładowe wywołania i wyniki programu

W niniejszym rozdziale zaprezentujemy zastosowania programu wraz z efektami dla różnych przypadków, aby zilustrować funkcjonowanie naszej aplikacji.

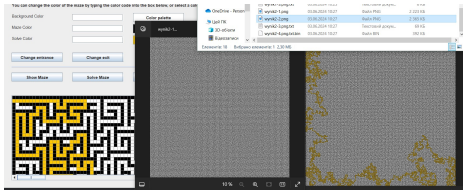
6.1 Podstawowe wywołanie dla pliku txt bez użycia dodatkowych funkcjonalności.

Pozwala pobrać rozwiązanie w plikach .txt i png Wynik:



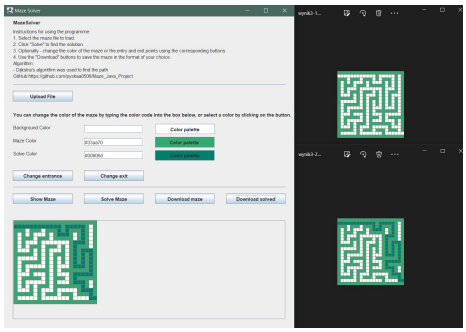
6.2 Podstawowe wywołanie dla pliku bin bez użycia dodatkowych funkcjonalności.

Pozwala pobrać rozwiązanie w plikach .txt, .bin i png Wynik:



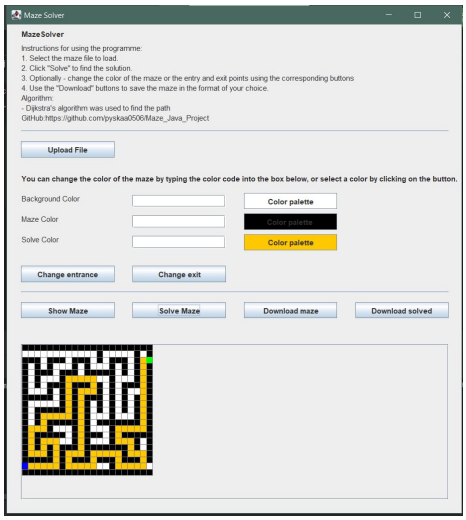
6.3 Wywołanie ze zmianą kolorów.

Wynik:



6.4 Wywołanie ze zmianą punktów wejścia i wyjścia

Wynik:



7 Zmiany względem specyfikacji

W początkowym założeniu rozwiązanie labiryntu miało być zapisywane wyłącznie w wymienionych wyżej formach dla pliku tekstowego i binarnego. Ostatecznie wymaganie to zostało rozszerzone o pobieranie obrazu labiryntu i jego ścieżki w formie png. Dostosowanie pod zmianę przebiegło płynnie, nie potrzebna była szczególna refaktoryzacja wcześniejszej części projektu.

8 Podsumowanie projektu

Projekt został zakończony z sukcesem, osiągając wszystkie założone cele. Dzięki szczegółowemu planowaniu i skrupulatnej implementacji, udało się stworzyć kompletny program do analizy labiryntów. Nowe moduły, takie jak możliwość wyboru kolorów, wprowadziły istotne ulepszenia w funkcjonalności. Projekt udało się wykonać na czas w całości.

9 Podsumowanie współpracy

Oczywiście: Współpraca przy projekcie labiryntów była płynna i efektywna. Regularny kontakt umożliwił szybkie dostosowanie się do zmian wprowadzonych do wymagań projektowych. Korzystanie z Git'a pozwoliło nam na równoczesną pracę nad projektem, eliminując stres. Nie było wiekszych problemów z komunikacją.

10 Wnioski

Projekt został pomyślnie zakończony, spełniając wszystkie założone cele. Opracowany program umożliwia skuteczną analizę labiryntów i generuje odpowiednie wyniki.
Dodanie nowych modułów, takich jak możliwość wyboru kolorów oraz możliwość pobrania rozwiązania w pliach .png, wniosło istotne ulepszenia w funkcjonalności programu, poszerzając jego możliwości.
Współpraca podczas projektu przebiegała płynnie i efektywnie, co pozwoliło na elastyczne dostosowanie się do zmian oraz uniknięcie zbędnego stresu.