Specyfikacja implementacyjna projektu *Wireworld* w języku Java

Chabik Jan (291060), Łuczak Mateusz (291088)

10 maja 2018

Spis treści

1 Informacje ogólne

Program jest aplikacją okienkową, w której wszystko obsługiwane jest z poziomu GUI. Okno będzie się pojawiać w centrum ekranu, a jego wielkość wynosić będzie 800x600. Użytkownik będzie miał możliwość wprowadzenia pliku wejściowego do programu w postaci macierzy o wybranej wielkości spośród 25x25, 50x50 i 100x100. Inne będą odrzucone przez program.

Samo GUI będzie zawierać przyciski umożliwiające użytkownikowi wybór pliku wejściowego, katalogu wyjściowego, atrybuty zapisu oraz uruchomienie automatu. Prócz wyżej wymienionych opcji interfejs udostępnia podgląd on-line przejść.

2 Opis klas i modułów

Wszystkie klasy znajdują się w pakiecie `com.wireworld`.

2.1 Klasa Wireworld

- Klasa wykonywalna tworząca GUI i wywołująca metodę obsługującą działanie programu.

2.2 Klasa Matrix

- Klasa zawierająca informacje o macierzy;
- Zawiera metody pozwalające na wypełnienie macierz obwodami i strukturami logicznymi.

2.3 Klasa Simulator

- Klasa abstrakcyjna z metodami statycznymi pozwalającymi symulować pojedyncze przejścia automatu *Wireworld*.

2.4 Klasa IOManager

- Klasa abstrakcyjna z metodami statycznymi pozwalającymi na sprawdzenie poprawności danych wejściowych, wczytanie ich do programu oraz utworzenie pliku `.txt` i `.png`.

2.5 Klasa Options

- Klasa typu *singleton* zawierająca informacje o opcjach podanych przez użytkownika;
- Dostępnymi opcjami są:
  - sąsiedztwo (Moore/von Neumann),
  - ścieżka pliku wejściowego,
  - ścieżka katalogu wyjściowego,
  - preferencje dotyczące zapisu (`.txt` i/lub `.png`),
  - informacja o interwale między przejściami automatu.
- Każda z opcji posiada swój *getter* i *setter*;
- Konstruktor jest prywatny i wywoływany podczas pierwszego odwołania się do metody zwracającej jedyną instancję klasy.

2.6 Klasa GUI

- Funkcje GUI są następujące:
  - wygenerowanie interfejsu użytkownika,
  - zapisywanie wprowadzonych przez użytkownika opcji do obiektu `options`,
  - umożliwienie użytkownikowi uruchomienie automatu i zarządzanie jego przejściami.

2.7 Klasa WireStructure

- Klasa zawierająca informacje dotyczące wielkości, położenia i orientacji bramek logicznych.

2.8 Klasa Wire

- Zawiera informacje o współrzędnych przewodnika.

2.9 Klasy odpowiadające za bramki logiczne (AND, OR, XOR, Diode)

- Klasa dziedzicząca po klasie *WireStructure* i stosująca polimorfizm.

3 Metodyka wersjonowania

- Wersjonowanie oprogramowania następuje w systemie kontroli wersji Git;
- Wersje dokumentów obarczone będą nazwą z dopiskiem `beta` lub `FINAL`, oraz numerem wersji:
  - `beta` - robocza wersja dokumentu;
  - `FINAL` - ostateczna wersja dokumentu;
  - numer wersji - numer oznaczający aktualną kompilację. Określony zostaje w formacie X.Y, gdzie X to numer kompilacji, a Y numer poprawki. Y jest opcjonalne i występuje tylko przy małych zmianach w dokumencie (takich jak przykładowo poprawa literówki).
- Wersje oprogramowania obarczone będą numerem wersji i dodatkową nazwą oznaczającą ważniejsze zmiany (przykładowo GUI v0.2)

4 Kompilator i wersja języka

Przy tworzeniu naszego oprogramowania używać będziemy języka Java w wersji 8 z kompilatorem `javac`. Zdecydowaliśmy się na tą wersję z powodu wygody użytkowania oraz dodatkowych opcji, które nie są dostępne we wcześniejszych wersjach (takich jak przykładowo wyrażenia `lambda`).

5 Użyte biblioteki

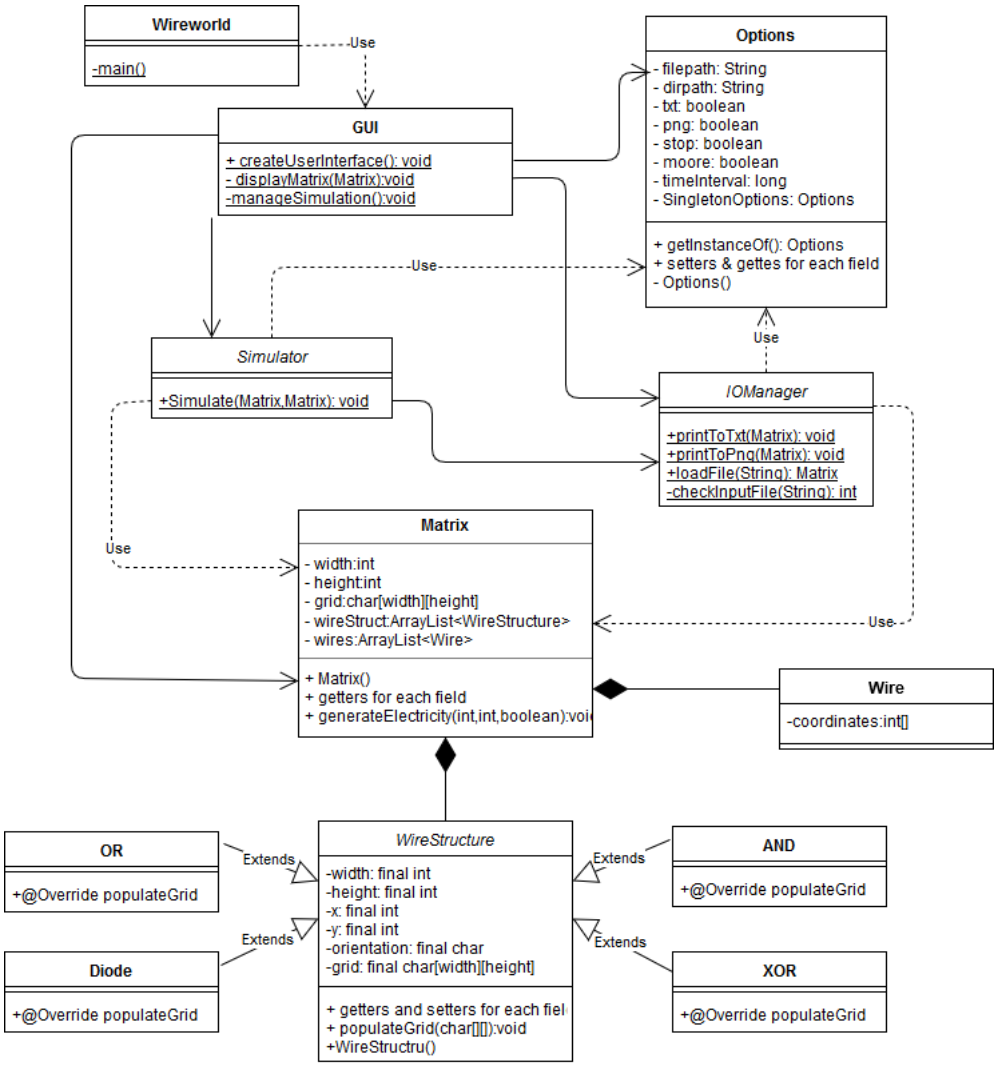
W naszym programie korzystamy z biblioteki Swing ze względu na łatwość obsługi, dodawania i modyfikacji działania wybranych komponentów, które zostaną użyte w GUI oprogramowania. Prócz Swinga korzystać będziemy ze standardowych bibliotek Javy dostarczanych wraz z *Java Delepoment Kit 1.8*.

6 Testowanie i konwencja

Testować działanie programu będziemy za pomocą narzędzi dostępnych w Frameworku JUnit 4. Dla każdej klasy stworzona zostanie klasa testująca, w której testować będziemy każdą z metod. Skorzystamy z metodyki testów regresywnych, czyli po dodaniu nowej funkcjonalności testować będziemy również poprzednio sprawdzone klasy w celu upewnienia się, czy program działa poprawnie.

7 Diagram klas

7.1 Diagram



7.2 Legenda

- Podkreślenie - Pola i metody statyczne
- *Kursywa* - Klasa abstrakcyjna