Specyfikacja implementacyjna automat komórkowy **"WireWorld"**

Bartosz Michałowski

1 kwietnia 2019

Spis treści

1 Informacje ogólne

1.1 Nazwa programu

Nazwa programu: `WireWorld`

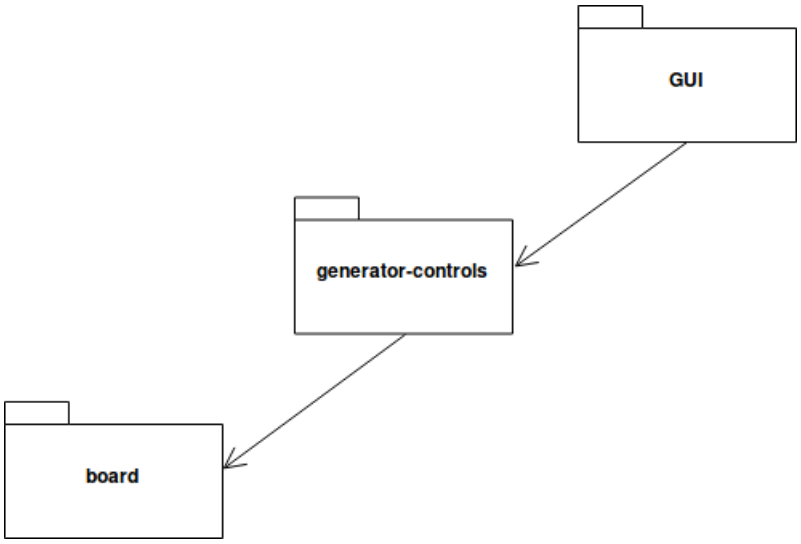
1.2 Język

Program zostanie napisany w języku **Java**. W programie nie przewiduje się zastosowania konstrukcji dostępnych w **Javie wersji 8** i wyższej.

1.3 Uruchomienie programu

Program przeznaczony jest do uruchamiania za pomocą pliku `WireWorld.jar`.

2 Diagram pakietów



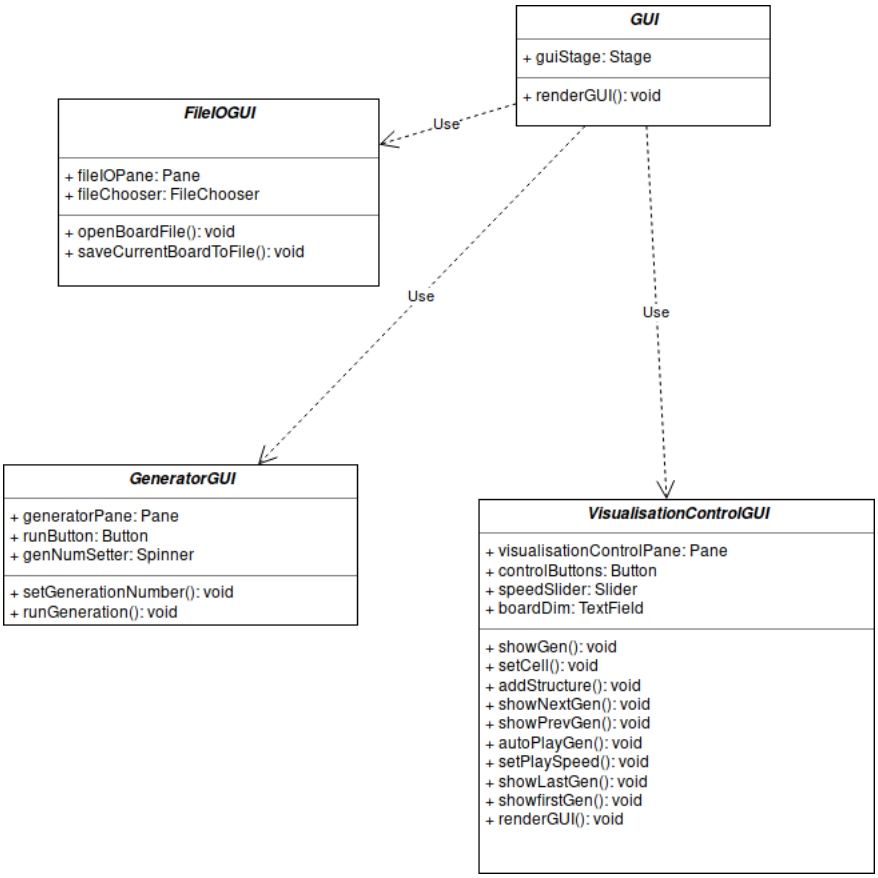
3 Opis pakietów

3.1 Pakiet `GUI`

3.1.1 Opis pakietu

Pakiet `GUI` zawiera klasy odpowiadające w całości za interfejs graficzny aplikacji. Klasą nadrzędną jest klasa GUI. Skorzystano z biblioteki JavaFX.

3.1.2 Diagram klas



3.1.3 Opis klas

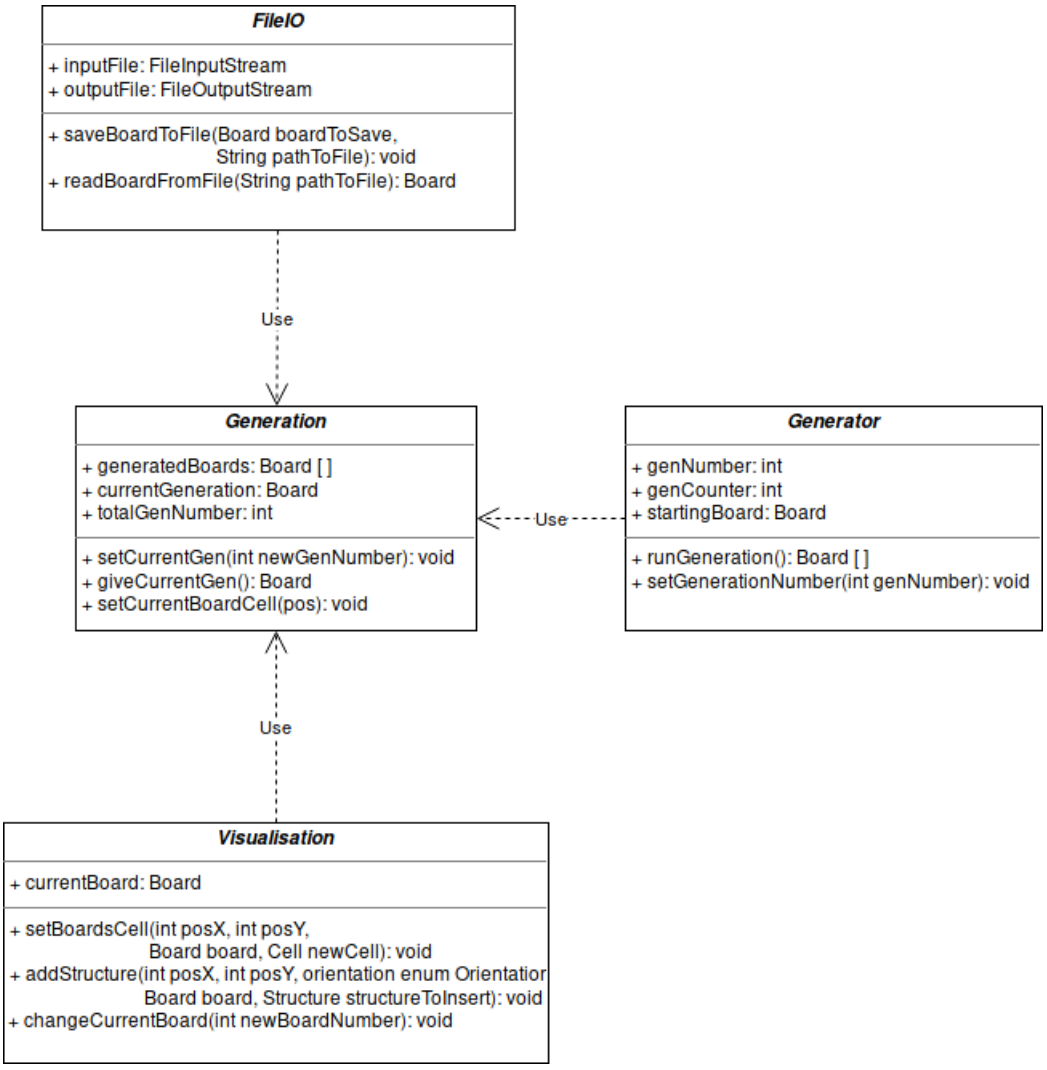
- Klasa `GUI` Jest to klasa opakowująca interfejs graficzny aplikacji w celu zapewnienia przejrzystości i ułatwienia tworzenia obsługi dla oddzielnych elementów programu.
- Klasa `FileIOGUI` Udostępnia użytkownikowi elementy sterujące pozwalające wczytać nową bądź zapisać obecną generację z/do pliku w formacie obsługiwanym przez aplikację.
- Klasa `GeneratorGUI` Udostępnia użytkownikowi dostęp do sterowania generatorem i modyfikację jego parametrów.
- Klasa `VisualisationGUI` Klasa generująca interfejs sterujący wizualizacją generacji, przechodzeniem między kolejnymi stanami, określanie prędkości przechodzenia itp.

3.2 Pakiet generation-controls

3.2.1 `generation-controls`

Pakiet `generation-controls` odpowiada za tworzenie kolejnych generacji planszy, możliwość eksportu ich do pliku, udostępnienie interfejsu dla bibliotek graficznych oraz za zapewnienie interfejsu umożliwiającego obsługę.

3.2.2 Diagram klas



3.2.3 Opis klas

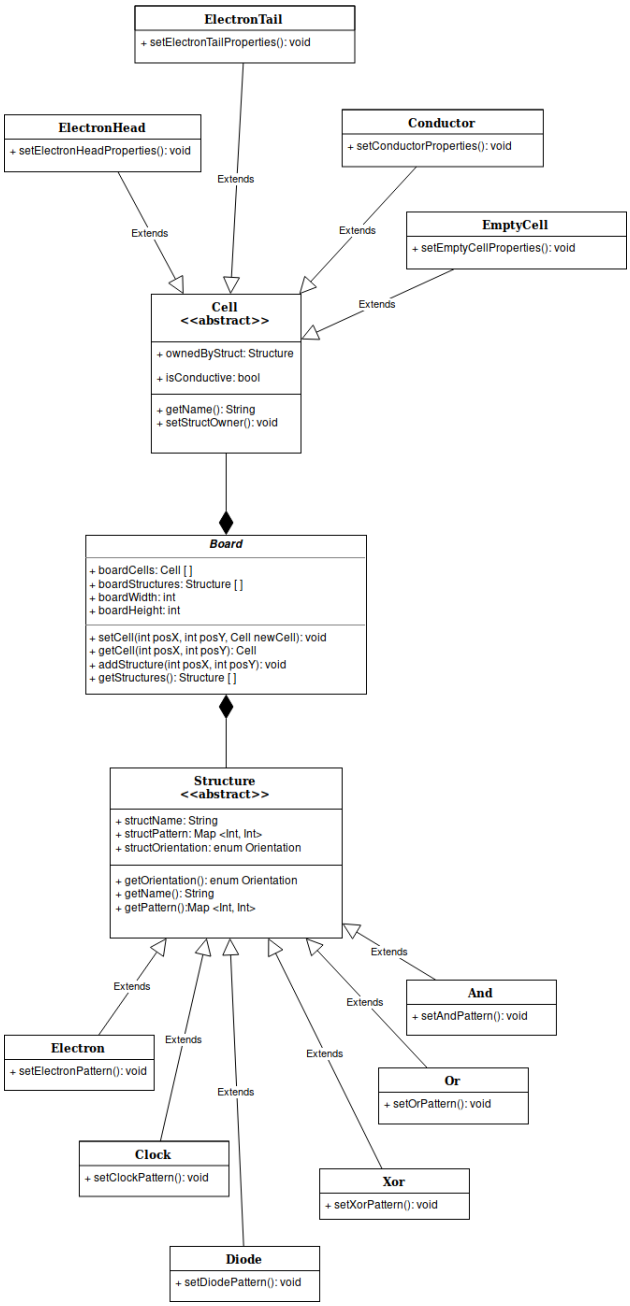
- Klasa `Generator` Moduł zawierający algorytm generacji kolejnych stanów automatu, modyfikujący obiekt klasy `Board` opisanej w dalszej części specyfikacji.
- Klasa `FileIO` Klasa umożliwiająca zapis i odczyt plików z generacjami zawierająca odpowiedni do tych celów interfejs dostępny dla innych klas.
- Klasa `Visualisation` Przetwarza obiekt klasy `Board` na obiekt możliwy do reprezentacji graficznej dla obiektu klasy `VisualisationGUI`
- Klasa `Generation` Przechowuje generacje wynikowe pochodzące od klasy `Generation`.

3.3 Pakiet`board`

3.3.1 Opis pakietu

Pakiet `board` zapewnia struktury niezbędne do przechowywania danych na temat generacji, jak i ich modyfikację oraz odczyt.

3.3.2 Diagram klas



3.3.3 Opis klas

- Klasa `Board` Klasa przechowująca wektor odpowiednich komórek oraz struktur generacji i udostępniająca interfejs pozwalający na ich odczytywanie i modyfikację.

- Klasa `Cell` Klasa abstrakcyjna reprezentująca pojedynczą komórkę i opakowująca ich różne rodzaje.
    1. Klasa `ElectronHead` Klasa rozszerzająca klasę `Cell` o komórkę przedstawiającą głowę elektronu.
    2. Klasa `ElectronTail` Klasa rozszerzająca klasę `Cell` o komórkę przedstawiającą ogon elektronu.
    3. Klasa `Conductor` Klasa rozszerzająca klasę `Cell` o komórkę symbolizującą przewodnik.
    4. Klasa `EmptyCell` Klasa rozszerzająca klasę `Cell` o pustą komórkę niebędącą przewodnikiem.
- Klasa `Structure` Klasa abstrakcyjna reprezentująca struktury składająca się z obiektów klas rozszerzających klasę `Cell` wraz z ich odpowiednimi pozycjami.
    1. Klasa `Electron` Klasa rozszerzająca klasę `Structure` o elektron składający się z głowy i ogona.
    2. Klasa `Diode` Klasa rozszerzająca klasę `Structure` o strukturę przepuszczającą elektrony tylko w jednym kierunku.
    3. Klasa `Clock` Klasa rozszerzająca klasę `Structure` o strukturę generującą periodycznie elektrony.
    4. Klasa `OR` Klasa rozszerzająca klasę `Structure` o bramke logiczną OR.
    5. Klasa `XOR` Klasa rozszerzająca klasę `Structure` o bramke logiczną XOR.
    6. Klasa `AND` Klasa rozszerzająca klasę `Structure` o bramke logiczną AND.

4 Przechowywanie danych w programie i metoda generacji

4.1 Przechowywanie danych w programie

Dane na temat generacji przechowywane są w obiekcie klasy `Generation` w postaci listy kolejnych stanów klasy `Board`, które z kolei zawierają informację na temat stanu komórek oraz zawartych w danej tablicy struktur. Przechowywanie informacji na temat struktur pozwala zapisać ich pozycje do pliku wyjściowego. W razie modyfikacji pola należącego do struktury, jest ona usuwana z generacji i dalej reprezentowana jedynie jako pola, które nie uległy modyfikacji.

4.2 Metoda generacji

Algorytm przechodzi po kolejnych polach obiektu klasy `Board` sprawdzając czy nie jest on polem pustym. Jeśli warunek ten jest spełniony pole modyfikowane jest na podstawie stanu komórek sąsiednich. Po przejściu całego obiektu jest on zapisywany jako kolejny stan w obiekcie klasy `Generation`.

5 Wymagania dotyczące użytkowania programu

Program jest przenośny, przez co rozumie się, że może zostać uruchomiony na każdym komputerze, który posiada zainstalowane `Java Runtime Environment (JRE)`, co w wolnym tłumaczeniu można określić jako *środowisko uruchamiania Java*. Jeżeli wyżej wymienione środowisko jest zainstalowane na danym komputerze to spełnia on wymagania systemowe odnośnie funkcjonowania programu. Program przeznaczony jest do uruchamiania za pomocą pliku `WireWorld.jar`.

6 Testowanie

Testowanie wodzące w projekcie to testowanie jednostkowe i użytkownika końcowego. Celem fazy testowania będzie sprawdzenie czy dana, tworzona przez nas funkcjonalność działa zgodnie z jej przeznaczeniem. Procesy testowania przeprowadzimy samodzielnie, głównie z użyciem narzędzi *JUnit*, *Mockito* i *AssertJ*.

6.1 Konwencja

Procesy testowania jednostkowego będą przebiegały z podstawowymi założeniami dobrych praktyk, będziemy tworzyć testy, które są niezależne od siebie, począwszy od najprostszych do najtrudniejszych funkcjonalności. Nazwy metod testujących będą starały się zobrazować oczekiwany efekt. Interfejs graficzny zostanie przetestowany ręcznie, sprawdzając czy interakcja z odpowiednimi jego elemantami zgadza się z przewidywanym rezultatem.

6.2 Narzędzia

Narzędzia użyte w procesie testowania to:

1. Biblioteka *JUnit*
2. Biblioteka *AssertJ*
3. Biblioteka *Mockito*

7 Wersjonowanie

Wersjonowanie projektu jest oparte o system kontroli wersji *Git*. Wersje programu są przechowywane w repozytorium `2018_JIMP2_repozytorium_gr1` stworzonym na potrzeby projektu. Kolejne wersje umieszczono w gałęzi `master` wyżej wymienionego repozytorium.

8 Narzędzia

Narzędzie użyte w procesie tworzenia programu to:

1. Zintegrowane środowisko deweloperskie *IntelliJ IDEA*
2. System kontroli wersji *Git*
3. Biblioteka *JUnit*
4. Biblioteka *AssertJ*
5. Biblioteka *Mockito*
6. Biblioteka *Maven*
7. Biblioteka *Apache Ant*
8. Biblioteka *JavaFX*