Specyfikacja implementacyjna
Automat komórkowy - WireWorld

Michał Sut

1 Informacje ogólne

1.1 Nazwa programu

Automat komórkowy *WireWorld*

1.2 Parametry początkowe programu

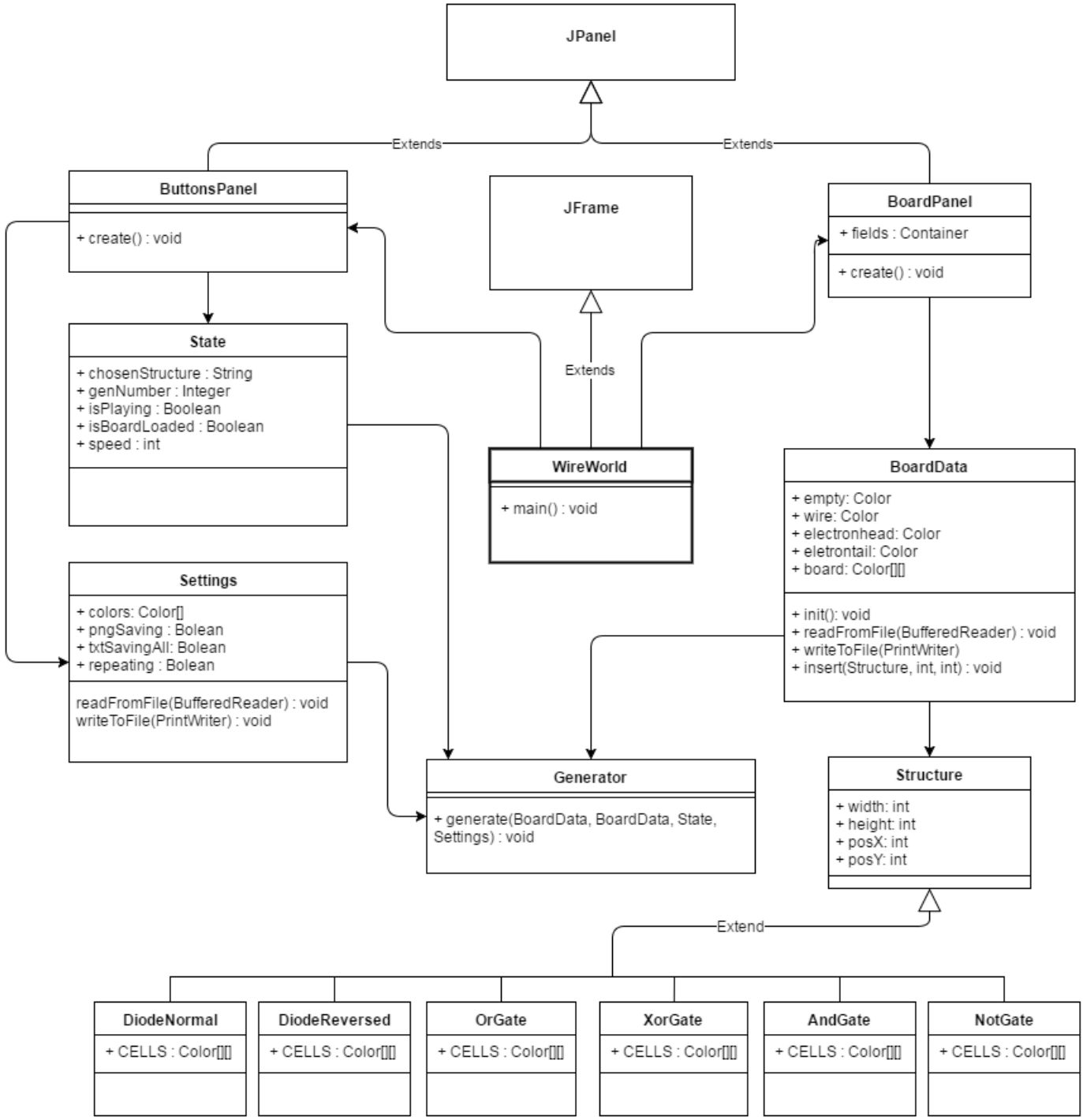
1. Okno programu ma stałą szerokość i wysokość – 970 x 600 pikseli.
2. Okno programu uruchamia się na środku ekranu.
3. Program rozpoczyna się bez wczytanej planszy. Przycisk **Play/Stop** jest nieaktywny. Aby móc rozpocząć, należy podać plik z danymi, bądź wprowadzić przynajmniej jeden element za pomocą interfejsu graficznego.

1.3 Działanie programu

Program, po wczytaniu planszy i uruchomieniu, wyświetla kolejne generacje w czasie rzeczywistym. Możliwa jest zmiana czasu upływającego pomiędzy wyświetleniami kolejnych generacji. Możliwe jest także przesuwanie generacji tylko o jeden etap. Opcja restartu powoduje usunięcie wszystkich struktur z planszy i przywrócenie programu do stanu przed wczytaniem danych.

2 Diagram klas

W celu zachowania czytelności diagramu, w klasach nie wypisano metod dostępowych, które służą jedynie pobieraniu lub ustawianiu wartości atrybutów.



Rysunek 1: Diagram klas.

3 Opis klas

3.1 WireWorld

Klasa *WireWorld* rozszerza klasę JFrame w celu utworzenia okna głównego aplikacji. Będzie ono uruchomione w metodzie **main** wraz z wcześniejszym nadaniem mu odpowiednich parametrów : wymiarów, położenia na ekranie oraz układu elementów (layout'u).

3.2 BoardPanel

Rozszerza klasę JPanel. Obiekt tej klasy jest panelem, na którym przechowywane będą pola automatu komórkowego w postaci mniejszych paneli. *BoardPanel* będzie miał układ siatki(Grid Layout), dzięki czemu elementy, które będzie zawierał ułożą się w dwuwymiarową planszę. Jedno pole takiej planszy to jedna komórka automatu komórkowego, które będzie zmieniało kolor w kolejnych generacjach zgodnie z zasadami automatu *WireWorld*. Odwołuje się do obiektu klasy *BoardData*, w której przechowywane są informację o stanie komórek.

3.3 BoardData

Klasa odpowiedzialna za przechowywanie stanu generacji. Obiekt tej klasy posiada tablicę dwuwymiarową wypełnioną kolorami, a także informacje o kolorach poszczególnych elementów (np. kolor głowy elektronu). Metody tej klasy odpowiadają za wczytanie danych o generacji początkowej z pliku, a także za wstawianie dowolnej struktury przy użycie interfejsu graficznego. Klasa ta powiązana jest z klasą *Structure*, Obiekty jej klas dziedziczących będą przekazywane do metody `insert` wstawiającej strukturę do planszy.

3.4 Structure i klasy dziedziczące

Structure jest klasą abstrakcyjną reprezentującą strukturę. Dziedziczą po niej klasy będące konkretnymi strukturami. Obiekty tych klas zawierają stałe tablice dwuwymiarowe wypełnione w sposób odpowiadający danemu elementowi.

3.5 ButtonsPanel

Rozszerza klasę JPanel i zawiera panel, w którym będą przechowywane przyciski. Wykorzystane będą dwa obiekty tej klasy. Pierwszy będzie przechowywał przyciski odpowiedzialne za sterowanie działaniem programu (np.: *Play/Stop*, *Faster, Slower*). Drugi panel będzie zawierał przyciski wstawiania struktur. Klasa *ButtonsPanel* wiąże się z klasami *State* oraz *Settings*, do których przekazuje informację o tym jakie przyciski zostały wciśnięte, po to by odpowiednio sterować działaniem programu (np.: wciskanie przycisku *Play/Stop* będzie wysyłało informację o tym czy program odtwarza generację czy jest zatrzymany.).

3.6 State

Obiekt tej klasy to stan aplikacji. Przechowuje informacje o tym, czy program jest w trakcie odtwarzania, jaka struktura będzie wstawiona po kliknięciu w planszę, jaka jest liczba generacji oraz odstęp czasowy między kolejnymi generacjami, a także czy na planszy znajdują się jakiekolwiek elementy. Obiekty tej klasy mogą służyć jako zbiór warunków definiujących zachowanie programu, ale także być wykorzystywane przy restarcie aplikacji, czyli przywróceniu danych z przed załadowania generacji.

3.7 Settings

Klasa, której obiekt przechowuje informacje o wybranych przez użytkownika ustawieniach. Ustawienia te będą zapisywane i odczytywane z pliku *settings.txt*. Pozwoli to na zapamiętanie ustawień przy wyjściu z aplikacji i odtworzenie ich podczas kolejnego uruchomienia. Wiąże się z klasą *Generator*, ponieważ przekuje do niej informację o tym, czy każda generacja ma być zapisywana do plików tekstowych lub graficznych.

3.8 Generator

Klasa odpowiadająca za przeprowadzanie generacji. Obiekt tej klasy to generator, który mając dwa obiekty klasy *BoardData* – „stary" i „nowy" – przekształca stan z pierwszego, „starego" obiektu, na „nowy" stan, a następnie zamienia obiekty, tak by „nowy" stał się „starym" i możliwe było wykonanie kolejnego kroku automatu komórkowego *WireWorld*. Z obiektów klas *State* i *Settings* czerpie informacje na temat tego, ile generacji ma wykonać, czy zapisywać każdą z nich do plików, jaki odstęp czasowy zastosować oraz czy zapętlić tworzenie generacji.

4 Testowanie

4.1 Testy poszczególnych części programu.

Testowanie polegało będzie na sprawdzeniu jak klasy lub zestawy klas realizują przeznaczone im zadania. W tym celu dla każdej klasy lub zestawu klas stworzony zostanie oddzielny, mniejszy program, który będzie wyświetlał wyniki działania danego modułu. Wyniki te sprawdzone zostaną pod kątem zgodności z oczekiwaniami.

4.1.1 BoardData

Klasa ta odpowiada za przechowywanie danych planszy oraz za wczytywanie i zapisywanie danych z/do pliku.
Metoda `readFromFile`, odpowiedzialna za wczytywanie danych, będzie miała za zadanie rozpoznawać pliki o niewłaściwym formacie i informować o tym użytkownika. Za niewłaściwy uznaje się plik zawierający: nierozpoznawalne nazwy struktur, liczby ujemne lub litery i znaki specjalne zamiast współrzędnych, współrzędne wychodzące poza rozmiar planszy.
Poprawność metody `writeToFile`, zapisującej dane do pliku, sprawdzona zostanie poprzez zbadanie, czy dane w pliku wyjściowym mają oczekiwany format.
Metoda `insert`, która wstawia elementy do planszy, zostanie wywołana dla kilku różnych struktur, a następnie sprawdzona zostanie zawartość tablicy dwuwymiarowej poprzez wyświetlenie jej na ekranie zastępując kolory liczbami.

4.1.2 Structure i klasy po niej dziedziczące

W celu przetestowania tych klas, ich obiekty zostaną wypisane na ekran, aby sprawdzić czy zawierają poprawne struktury. Zostanie również zbadane działanie metod pobierających wartości atrybutów (tzw. getterów) poprzez wypisanie na ekran pobranych wartości.

4.1.3 Generator

Obiekt klasy *Generator* ma za zadanie zmieniać stan planszy zawarty w obiektach klasy *BoardData*, uwzględniając informacje z obiektów klas *State* oraz *Settings*. Przetestowanie tej klasy w odosobnieniu od innych będzie oznaczało wykonanie metody `generate` dla pewnych ręcznie utworzonych tablic dwuwymiarowych i stałych wartości prędkości generacji oraz kolorów. Na ekran wypisane zostaną dwa etapy generacji, aby sprawdzić, czy stan pierwszy został odpowiednio przekształcony w stan kolejny.

4.2 State, Settings

Po wciśnięciu przycisków, które zmieniają ustawienia lub stan aplikacji, na ekran zostaną wypisane aktualne wartości zmiennych przechowujących stan programu lub ustawienia. Następnie będą one porównane z wartościami oczekiwanymi.

4.2.1 ButtonPanel, BoardPanel

Klasy te odpowiadają za graficzny interfejs programu. Sprawdzony zostanie układ przycisków oraz planszy w oknie aplikacji, a także ich zachowanie po kliknięciu. Komórki planszy, przechowywane w obiekcie klasy *BoardPanel*, powinny zmieniać kolor, a przyciski w *ButtonPanel* powinny zmieniać wartości zmiennych stanu.

4.2.2 WireWorld

Klasa ta zawiera jedną metodę: `main`, w której utworzone i otworzone zostanie główne okno programu. Test polegał będzie jedynie na sprawdzeniu czy okno otwiera się prawidłowo, tzn. z ustalonymi parametrami startowymi, takimi jak: szerokość, wysokość, położenie na ekranie.

4.3 Testy ogólnego działania programu

Po przetestowaniu poszczególnych części programu, zbadane zostanie działanie aplikacji pod względem realizacji zasad automatu WireWorld. Będzie to polegało na porównaniu wyników generacji z wynikami zewnętrznej aplikacji. Do tego celu użyte zostanie narzędzie *WireWorld - online*. Sprawdzane będzie, czy kolejne etapy generacji są takie same.
Przetestowane zostanie, czy program działa zgodnie z zasadami dla plików z różnymi generacjami, również takich, które nie są wypełnione wcale albo zapełniają całą planszę.