Politechnika Warszawska
Wydział Elektryczny

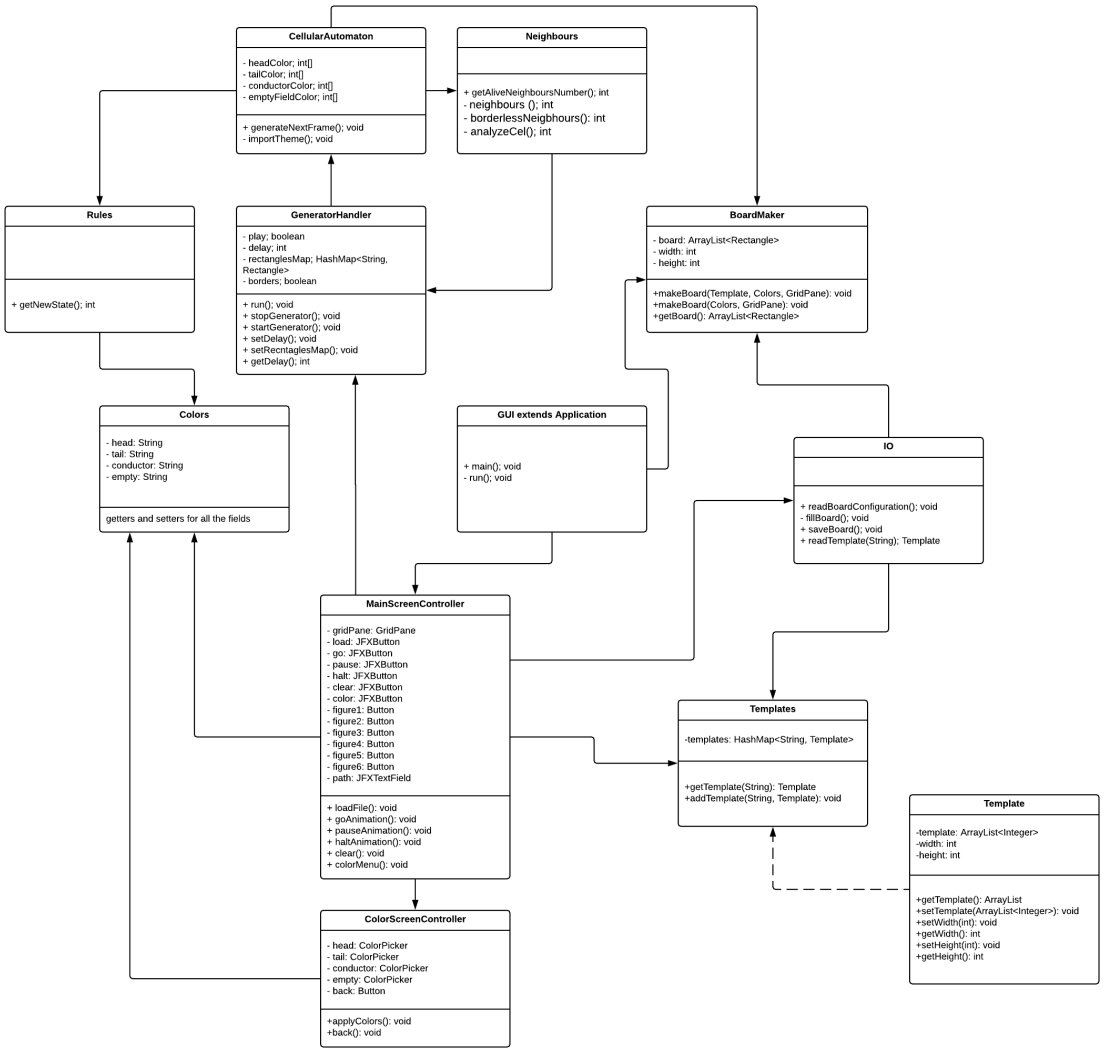
Specyfikacja Implementacyjna "WireWorld"

Autorzy:
Grzegorz Kopyt
Daniel Sporysz

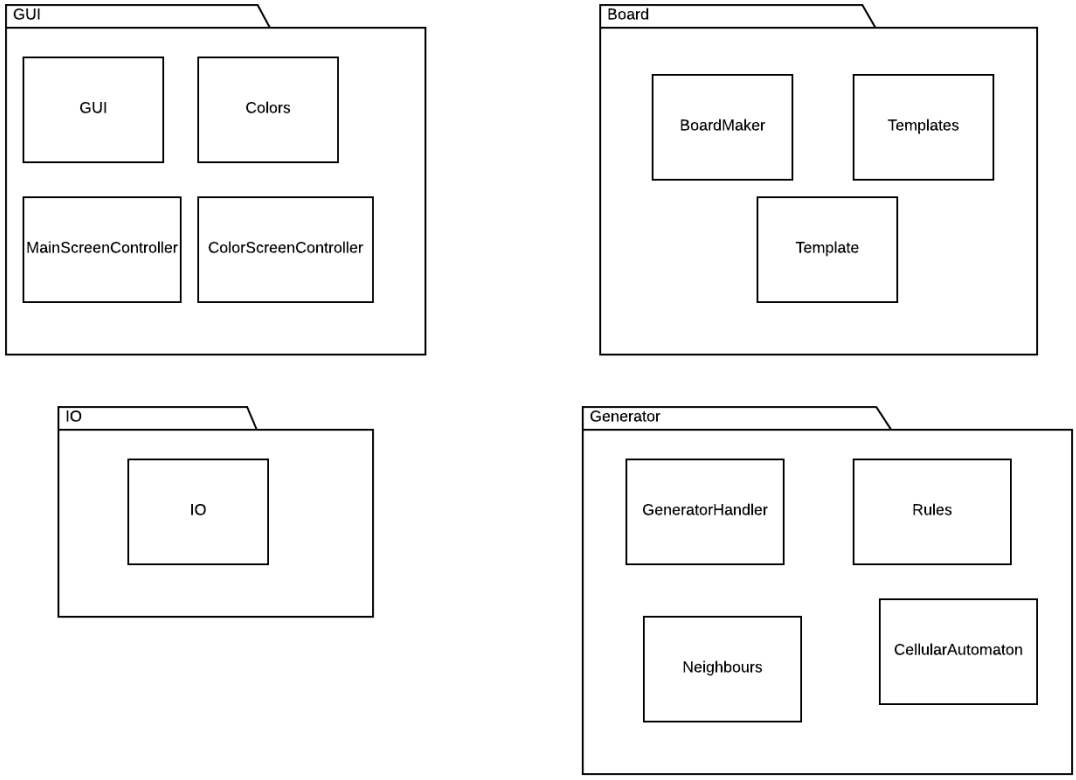
12 maja 2018

Spis treści

1 Diagram klas



2 Diagram pakietów



---

3 Pakiet GUI

Wykonany przy pomocy technologii javafx, zawiera dodatkową bibliotekę: jfoenix.

3.1 Pliki .fxml:

- MainScreen.fxml
- ColorScreen.fxml

3.2 GUI

Rozszerza klasę Application z javafx.

3.2.1 Pola

brak

3.2.2 Metody

- **main**

  Standardowo wywołuje metodę launch.

- **start**

  Wczytuje plik MainScreen.fxml, przygotowuje całą scenę i wyświetla ją w wymiarach (800, 600).

3.3 MainScreenController

Kontroler sceny MainScreen.fxml.

3.3.1 Pola

- GridPane *gridPane*
- JFXButton *load*
- JFXButton *go*
- JFXButton *pause*
- JFXButton *halt*
- JFXButton *clear*
- JFXButton *color*
- Button *figure1*
- Button *figure2*
- Button *figure3*
- Button *figure4*
- Button *figure5*
- Button *figure6*
- JFXTextField *path*

3.3.2 Metody

- void **loadFile()**
  Standardowo wywołuje metodę launch.
- void **goAnimation()**
  Uruchamia animacje wywołując metodę z klasy Animation.
- void **pauseAnimation()**
  Pauzuje animacje metodą z klasy Animation.
- void **haltAnimation()**
  Powoduje powrót animacji do punktu początkowego.
- void **clear()**
  Zmienia kolor każdej komórki w tablicy na biały.
- void **colorMenu()**
  Wyświetla okno z pliku ColorMenu.fxml

3.4 ColorScreenController

Kontroler sceny ColorScreen.fxml.

3.4.1 Pola

- ColorPicker *head*
- ColorPicker *tail*
- ColorPicker *conductor*
- ColorPicker *empty*

3.4.2 Metody

- void **back()**

  Powoduje powrót do MainScreen.

- void **applyColors()**

  Pobiera z obiektów klasy ColorPicker informacje o kolorach i przekazuje je klasie Colors.

3.5 Colors

Kontroler sceny ColorScreen.fxml.

3.5.1 Pola

- String head
- String tail
- String conductor
- String empty

3.5.2 Metody

- String **getHead()**
- void **setHead()**
- String **getTail()**
- void **setTail()**
- String **getConductor()**
- void **setConductor()**
- String **getEmpty()**
- void **setEmpty()**

3.5.3 Konstruktor

Domyślne kolory to:

- head - żółty
- tail - czerwony
- conductor - czarny
- empty - biały

---

4 Pakiet IO

4.1 IO

krótki opis klasy

4.1.1 Pola

4.1.2 Metody

---

5 Pakiet Board

5.1 BoardMaker

Odpowiada za stworzenie tablicy obiektów klasy Rectangle (*board*). Tablica ta będzie służyła jako obszar edytowany przez użytkownika, a także będzie na niej wyświetlana animacja.

5.1.1 Pola

- ArrayList<Rectangle> *board*
- int *width*
- int *height*

5.1.2 Metody

- void **makeBoard(Template, Colors, GridPane)**

  Metoda tworzy tablicę *board* obiektów klasy Rectangle na podstawie wzoru podanego w Template o 30 kwadratach w rzędzie i 30 kwadratach w kolumnie. Wymiary obiektów Rectangle wynoszą (20, 20). Wszystkie obiekty dodane zostają do kolekcji *board*. Obiektom zostaje nadane ID jako kolejne liczby naturalne całkowite zaczynając od 1. Kolor wypełnienia obiektów nadawany jest zgodnie z zawartością Colors, obramowanie - czarne. Tablica tworzona jest poprzez dodanie obiektów do GridPane. Docelowy GridPane znajduje się w klasie MainScreenController.

- void **makeBoard(Colors, GridPane)**

  Metoda tworzy tablicę *board* obiektów klasy Rectangle o 30 kwadratach w rzędzie i 30 kwadratach w kolumnie. Wymiary obiektów Rectangle wynoszą (20, 20). Wszystkie obiekty dodane zostają do kolekcji *board*. Obiektom zostaje nadane ID jako kolejne liczby naturalne całkowite zaczynając od 1. Kolor wypełnienia obiektów nadawany jest zgodnie z zawartością Colors, obramowanie - czarne. Tablica tworzona jest poprzez dodanie obiektów do GridPane. Docelowy GridPane znajduje się w klasie MainScreenController.

- ArrayList<Rectangle> **getBoard()**

5.2 Template

Klasa reprezentująca wzór obiektu do wstawiania na tablice *board*. Wzór przechowywany jest w tablicy *template* jako ciąg liczb 0(pusty), 1(przewodnik), 2(ogon), 3(głowa). Przy tworzeniu wzorów należy uwzględnić to, że rozmiar tablicy wynosi 30 kwadratów na 30 kwadratów.

5.2.1 Pola

- ArrayList<Integer> *template*
- int *width*
- int *height*

5.2.2 Metody

- Template **getTemplate()**
- void **setTemplate(ArrayList<Integer>)**
- void **setWidth(int)**
- int **getWidth()**
- void **setHeight(int)**
- int **getHeight()**

5.3 Templates

Przeznaczeniem klasy jest przechowywanie obiektów klasy template w kolekcji.

5.3.1 Pola

- HashMap<String, Template> *templates*

5.3.2 Metody

- Template **getTemplate(String)**

  Metoda otrzymawszy klucz klasy *String* zwraca obiekt klasy Template z kolekcji *templates*.

- void **addTemplate(String, Template)**

  Metoda dodaje do kolekcji *templates* obiekt klasy Template i nadaje mu klucz podany jako zmienna klasy String.

---

6 Pakiet Generation

6.1 GenerationsHandler

krótki opis klasy

6.1.1 Pola

- typ *nazwa*

6.1.2 Metody

- typ **nazwa**

  opis metody

6.2 CellularAutomaton

krótki opis klasy

6.2.1 Pola

- typ *nazwa*

6.2.2 Metody

- typ **nazwa**
  opis metody

6.3 Rules

krótki opis klasy

6.3.1 Pola

- typ *nazwa*

6.3.2 Metody

- typ **nazwa**
  opis metody

---

7 Przepływ Sterowania

---

8 Testy klas

8.1 GUI

8.1.1 MainScreenController

Scenariusze

1.
2.
3.
4.
5.

Kryteria oceny poprawnej pracy

1.
2.
3.
4.
5.

8.2 IO

8.2.1 IO

Scenariusze

1.
2.
3.
4.
5.

Kryteria oceny poprawnej pracy

1.
2.
3.
4.
5.

8.3 Board

8.3.1 BoardMaker

Scenariusze

1.
2.
3.
4.
5.

Kryteria oceny poprawnej pracy

1.
2.
3.
4.
5.

8.4 Generation

8.4.1 GenerationsHandler

Scenariusze

1.
2.
3.
4.
5.

Kryteria oceny poprawnej pracy

1.
2.
3.
4.
5.