

Politechnika Warszawska  
Wydział Elektryczny

---

SPECYFIKACJA IMPLEMENTACYJNA  
"WIREWORLD"

---

*Autorzy:*  
GRZEGORZ KOPYT  
DANIEL SPORYSZ

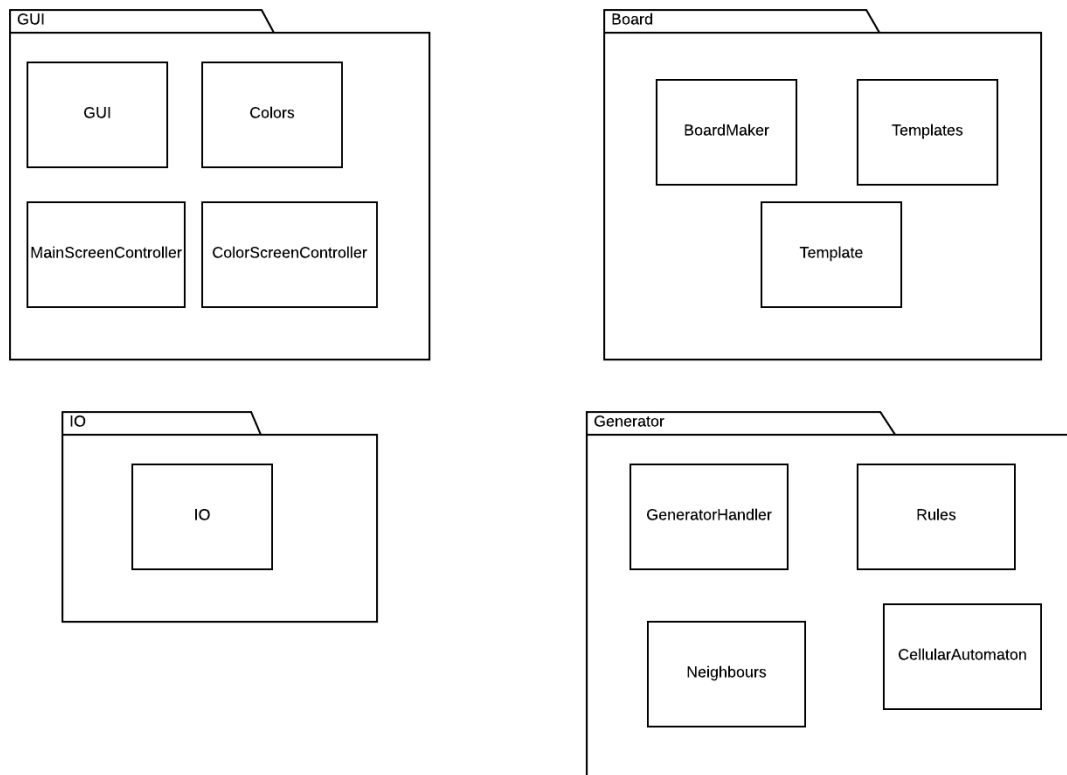
12 maja 2018

# Spis treści

<b>1</b>	<b>Diagram klas</b>	<b>2</b>
<b>2</b>	<b>Diagram pakietów</b>	<b>3</b>
<b>3</b>	<b>Pakiet GUI</b>	<b>3</b>
3.1	Pliki .fxml: . . . . .	3
3.2	GUI . . . . .	3
3.2.1	Pola . . . . .	3
3.2.2	Metody . . . . .	3
3.3	MainScreenController . . . . .	4
3.3.1	Pola . . . . .	4
3.3.2	Metody . . . . .	4
3.4	ColorScreenController . . . . .	4
3.4.1	Pola . . . . .	5
3.4.2	Metody . . . . .	5
3.5	Colors . . . . .	5
3.5.1	Pola . . . . .	5
3.5.2	Metody . . . . .	5
3.5.3	Konstruktor . . . . .	5
<b>4</b>	<b>Pakiet IO</b>	<b>6</b>
4.1	IO . . . . .	6
4.1.1	Pola . . . . .	6
4.1.2	Metody . . . . .	6
<b>5</b>	<b>Pakiet Board</b>	<b>6</b>
5.1	BoardMaker . . . . .	6
5.1.1	Pola . . . . .	6
5.1.2	Metody . . . . .	6
5.2	Template . . . . .	6
5.2.1	Pola . . . . .	7
5.2.2	Metody . . . . .	7
5.3	Templates . . . . .	7
5.3.1	Pola . . . . .	7
5.3.2	Metody . . . . .	7
<b>6</b>	<b>Pakiet Generation</b>	<b>7</b>
6.1	GenerationsHandler . . . . .	7
6.1.1	Pola . . . . .	7
6.1.2	Metody . . . . .	7
6.2	CellularAutomaton . . . . .	8
6.2.1	Pola . . . . .	8
6.2.2	Metody . . . . .	8
6.3	Rules . . . . .	8
6.3.1	Pola . . . . .	8
6.3.2	Metody . . . . .	8
<b>7</b>	<b>Przepływ Sterowania</b>	<b>8</b>



## 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()**  
Wywołuje metodę, która wczytuje plik tekstowy, którego ścieżkę podano w polu tekstowym.
- 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 tablicę *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

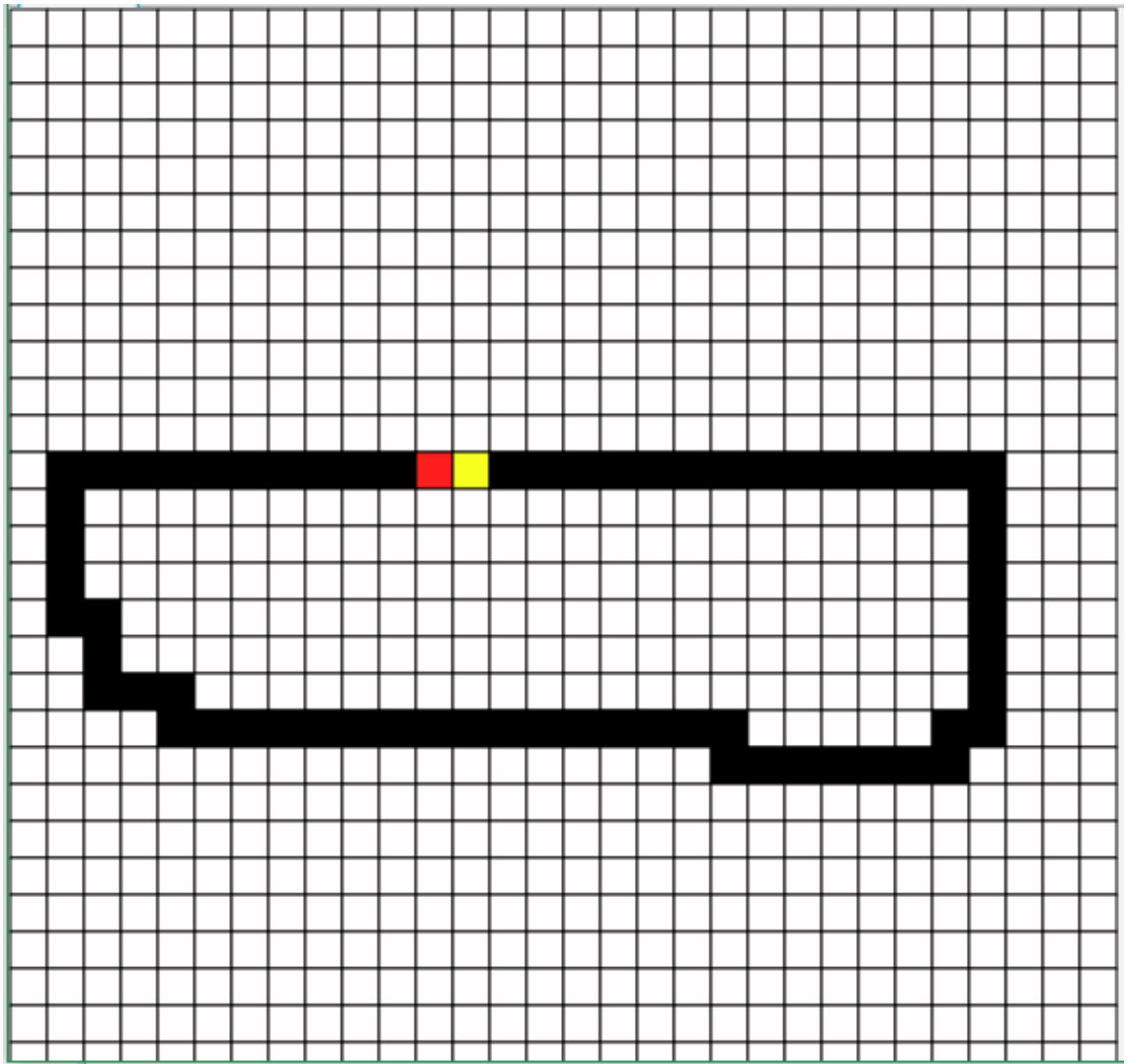
---

## 8 Testy klas i GUI

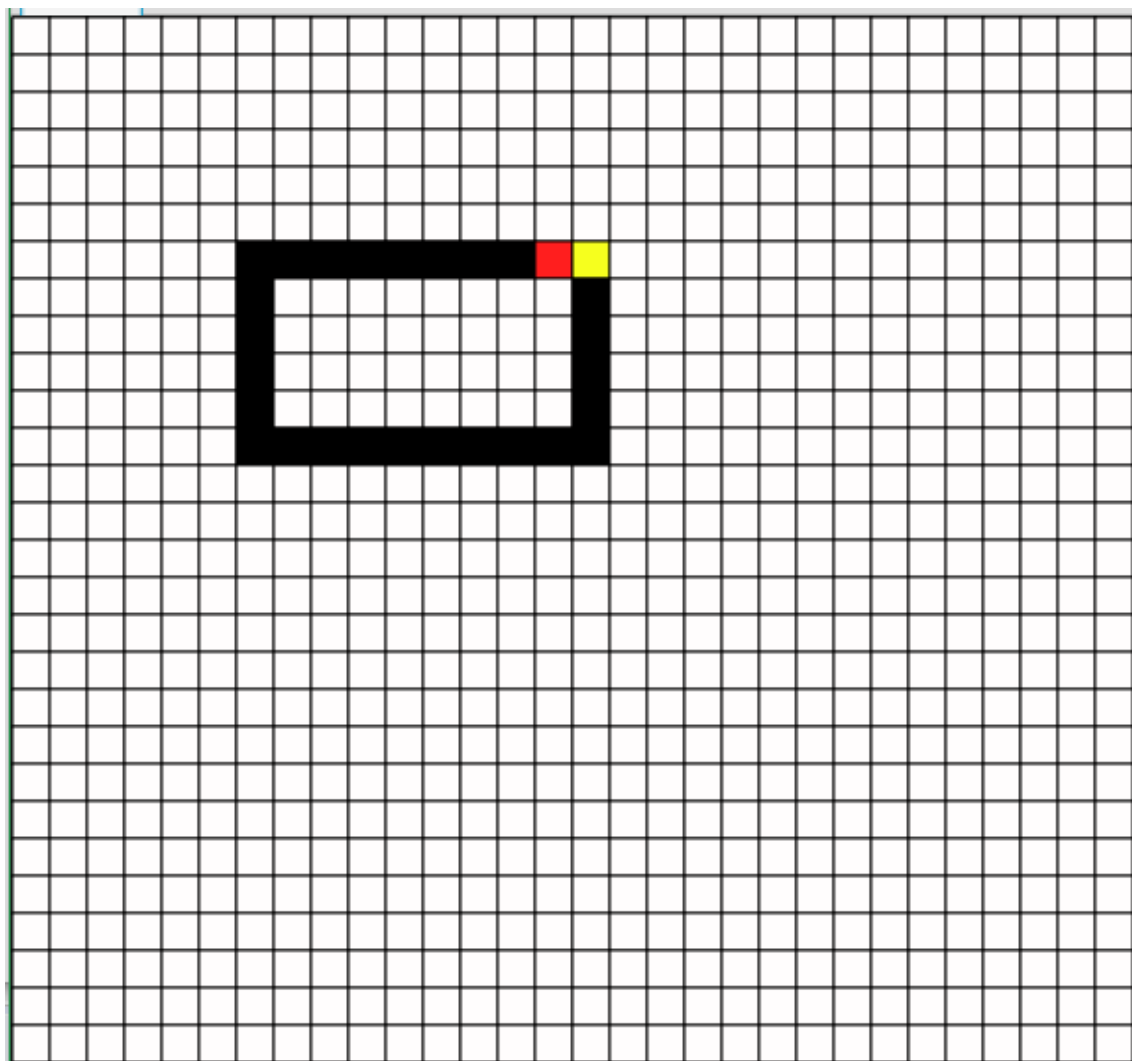
### 8.1 GUI

#### Scenariusze

1. Należy wpisać w pole tekstowe poprawną ścieżkę pliku (TestLoad.txt) i kliknąć w przycisk Load.
2. Należy wpisać w pole tekstowe niepoprawną ścieżkę pliku i kliknąć w przycisk Load.
3. Należy wpisać w pole 101 znaków.
4. Należy wpisać w pole tekstowe „....!fte./tetat/assfs///// ”.
5. Należy najechać myszką na planszę edycji.
6. Należy czterokrotnie kliknąć na dowolne pole planszy.
7. Należy przejechać po planszy ze wciśniętym lewym przyciskiem myszy.
8. Należy wyklikać na planszy taki wzór:

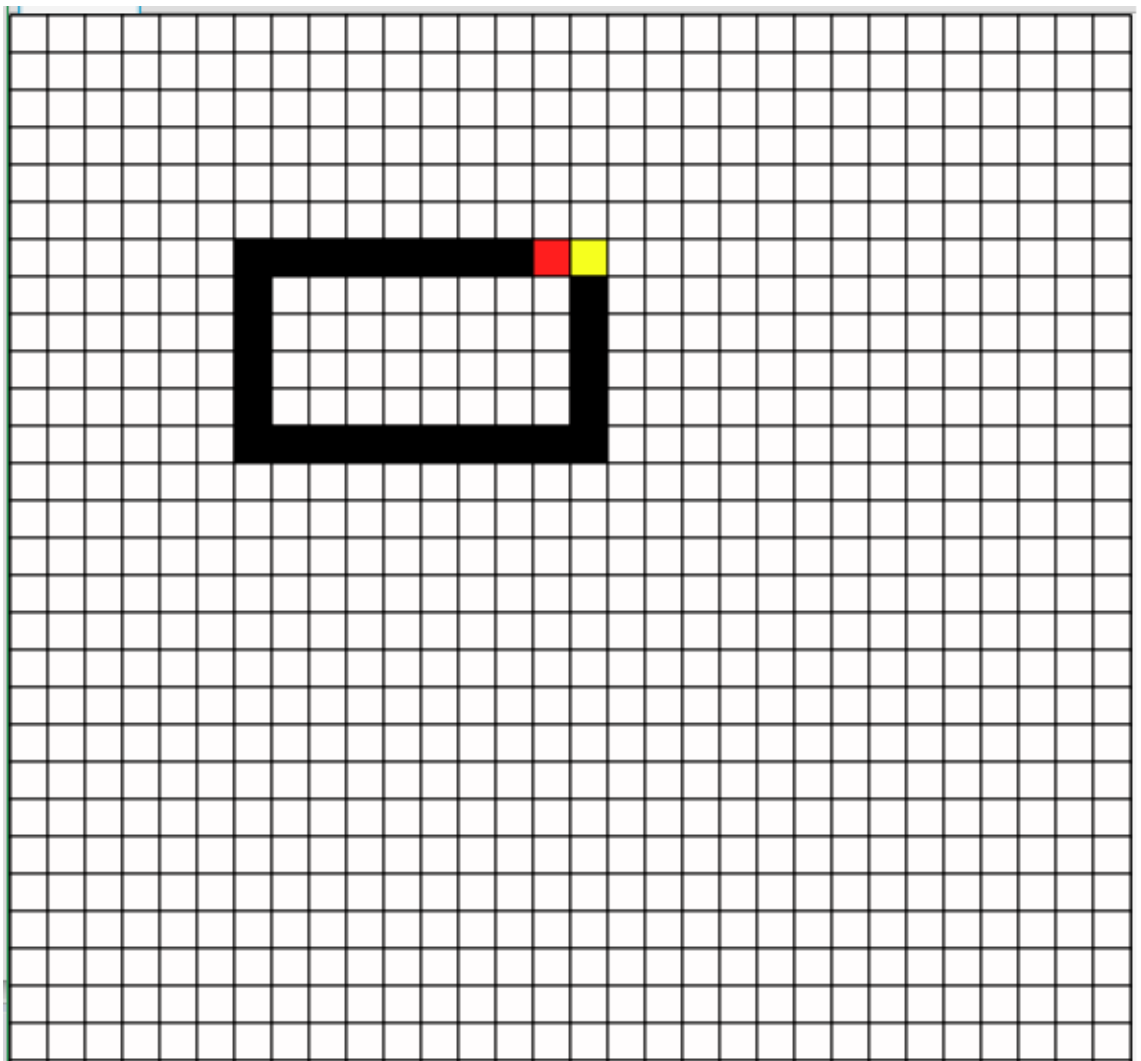


9. Po wykonaniu poprzedniego testu należy wcisnąć przycisk GO.
10. Po wykonaniu poprzedniego testu należy wcisnąć przycisk Pause.
11. Po wykonaniu poprzedniego testu należy wcisnąć przycisk Halt!.
12. Należy kliknąć w przycisk Clear.
13. Kiedy na planszy są same białe komórki należy wcisnąć każdy z przycisków GO, Pause, Halt!
14. Dla każdego przycisku figure, należy: kliknąć na niego, następnie kliknąć na planszy w komórkę o współrzędnych (15,15) i komórkę (30,15)
15. Należy kliknąć w przycisk Color.
16. Należy ustawić plansze w sposób przedstawiony na obrazku poniżej. W oknie ColorMenu ustawić cztery różne kolory, następnie kliknąć przycisk back i kliknąć przycisk GO.

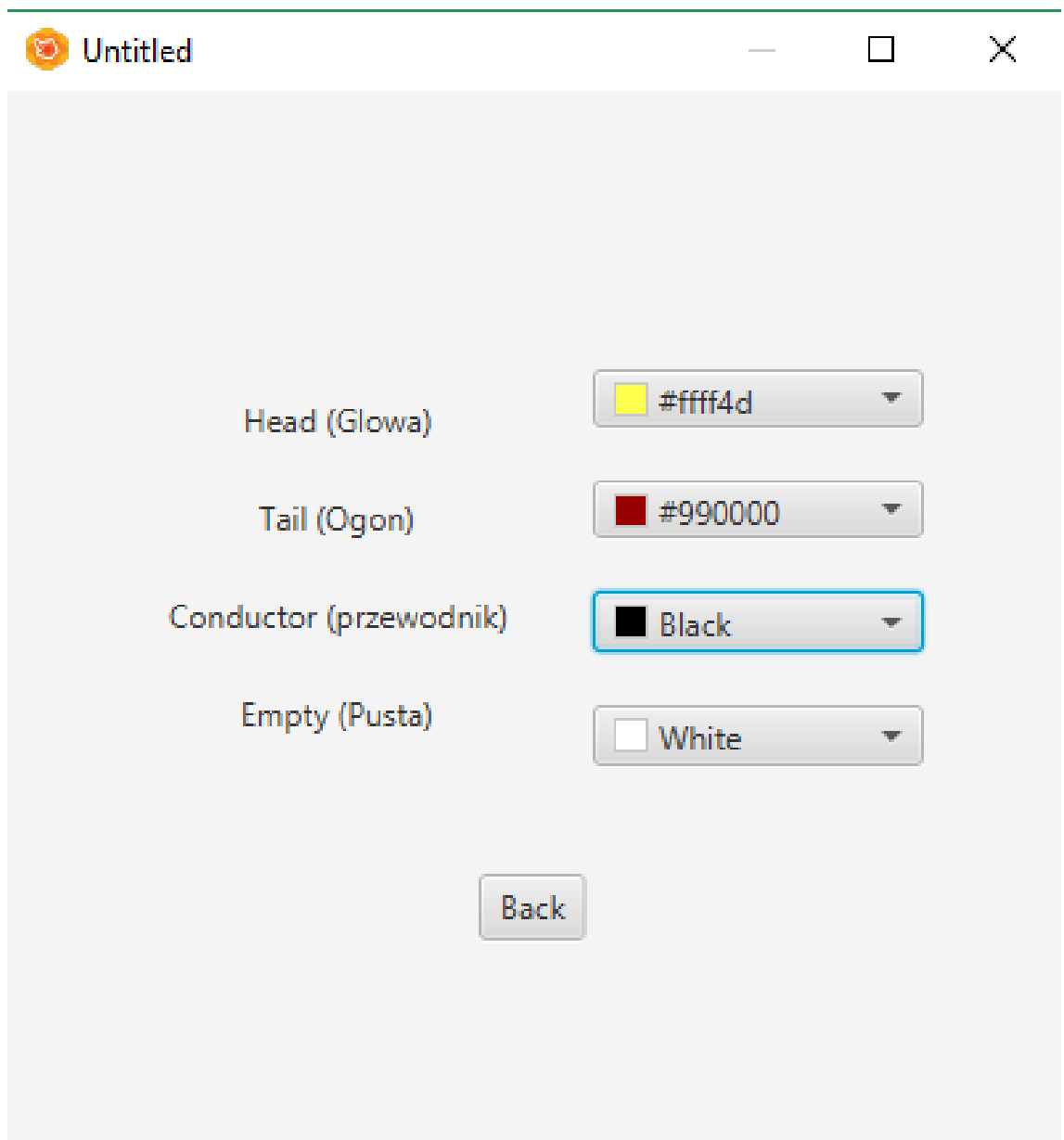


### Kryteria oceny poprawnej pracy

1. Jeśli na planszy pojawi się taki układ pól jak poniżej, test jest zaliczony.



2. Program powinien wyświetlić w polu tekstowym czerwony komunikat "Can't open the file."
3. Program powinien wyświetlić w polu tekstowym czerwony komunikat "Can't open the file."
4. Program powinien pozwolić na wpisanie tylko 100 znaków.
5. Kolory nie powinny się zmieniać.
6. Kolory powinny zmienić się cztery razy (czarny->czerwony->żółty->biały).
7. Kolory powinny się zmienić o jeden.
8. Jeśli udało się to test jest zdany.
9. Kolory powinny się zmieniać zgodnie z zasadami wire world.
10. Animacja powinna się zatrzymać.
11. Animacja powinna wrócić do początkowego układu sprzed kliknięcia przycisku GO.
12. Wszystkie komórki powinny być białe.
13. Kolory na planszy nie powinny się zmieniać.
14. Elementy powinny zostać wstawione na planszę, z wyjątkiem przypadku, w którym zahaczają o krawędź ekranu.
15. Powinno pojawić się ColorMenu:



16. Kolory na planszy powinny zostać zmienione zgodnie z ustawieniami, a animacja odbyć się zgodnie z zasadami wire world.

## 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

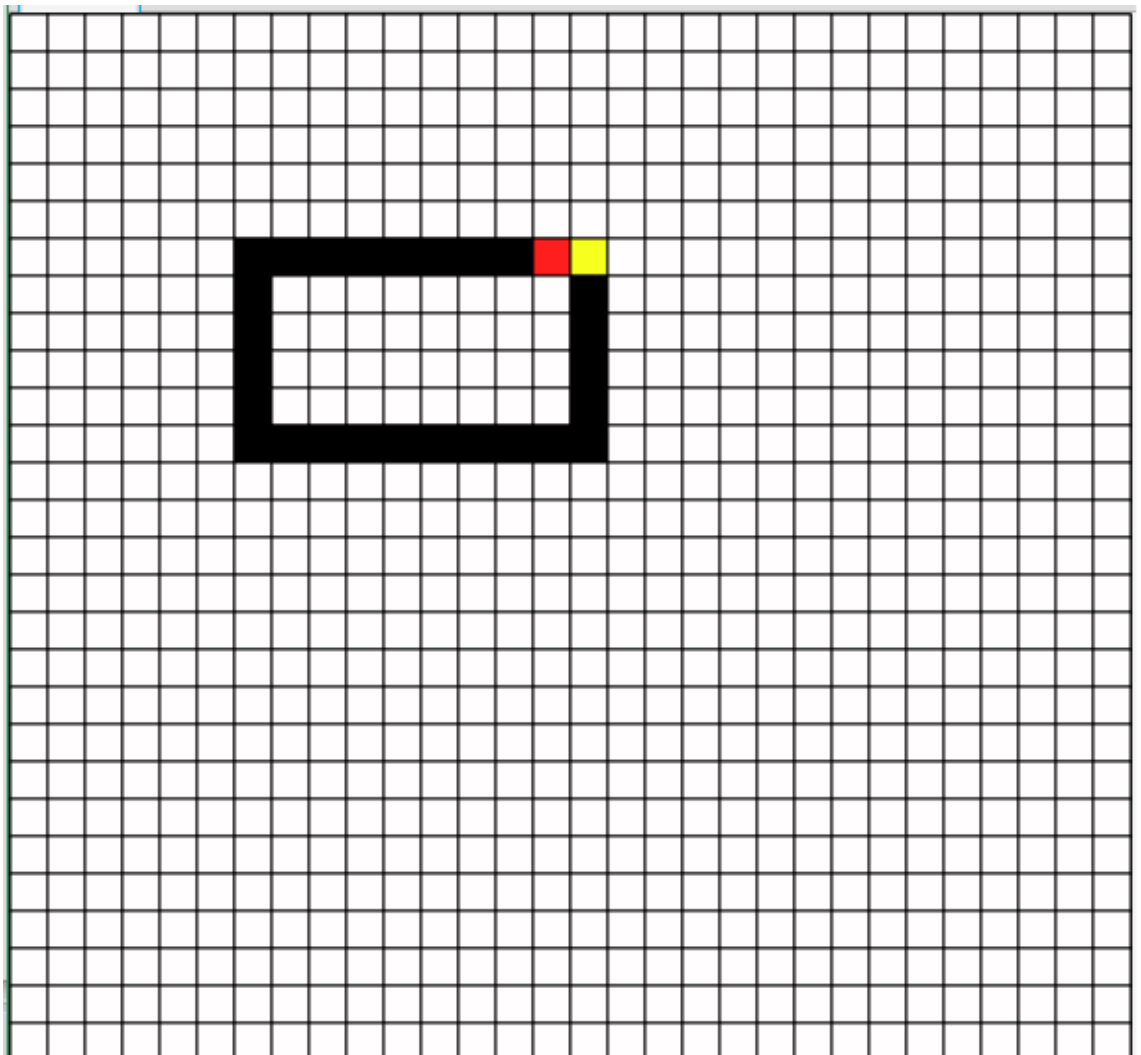
### 8.3.1 BoardMaker, Template

#### Scenariusze

1. Stworzyć obiekt Template jako wartości pól wymiarów podać (30, 30), a listę obiektów klasy Integer taką jak w pliku TestLoad.txt. Wypisać macierz na konsole.
2. Wywołać metode makeBoard w wariacie bez argumentu Template. Podając domyślny GridPane i obiekt Colors z domyślnymi kolorami.
3. Wywołać metode makeBoard w wariacie z argumentem Template takim jak w teście pierwszym. Podając domyślny GridPane i obiekt Colors z domyślnymi kolorami.

#### Kryteria oceny poprawnej pracy

1. Jeśli macierz jest taka sam jak w pliku TestLoad.txt, test jest zdany.
2. W głównym menu interfejsu graficznego powinna pojawić się tablica białych kwadratów 30 na 30.
3. W głównym menu interfejsu graficznego powinna pojawić się taka tablica:



### 8.3.2 Templates

#### Scenariusze

1. Stworzyć puste obiekty `Template`, których pola `x` wypełnić kolejno wartościami 1, 2, 3. Dodać te obiekty do `Tempaltes` za pomocą metody *`addTempalte`* nadając im klucze 1, 2 ,3. Pobrać z `Templates` obiekty `Template` po kolejno 1, 2, 3 nadanych kluczach i wypisać wartości ich pól `x` na konsoli.

#### Kryteria oceny poprawnej pracy

1. Na konsoli powinien pojawić się komunikat: „1 2 3”

### 8.4 Generation

#### 8.4.1 GenerationsHandler

#### Scenariusze

- 1.
- 2.
- 3.
- 4.
- 5.

#### Kryteria oceny poprawnej pracy

- 1.
- 2.
- 3.
- 4.
- 5.