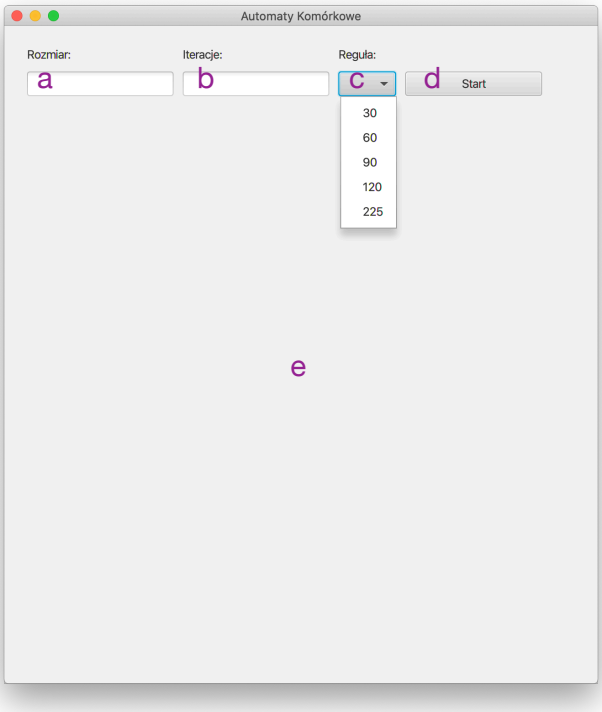
Weronika Smieja IS, rok III, gr. 4. Lab. 7. Automaty komórkowe 26. 03. 2020 r.

Git: **wsmieja** Repozytorium: **https://github.com/KISiM-AGH/zca1-wsmieja**

Sprawozdanie

1. Wykonanie - interfejs aplikacji



Rys. 1. Interfejs

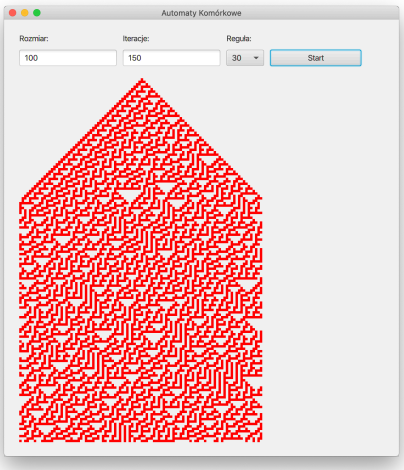
Ćwiczenie wykonano w Javie, do stworzenia GUI użyto JavaFX.
Interfejs aplikacji, przedstawiony na rys. 1., zawiera następujące elementy:

a) Pole tekstowe, umożliwiające użytkownikowi wybór rozmiaru przestrzeni;
b) Pole tekstowe do wyboru liczby iteracji
c) Pole wyboru z listą obsługiwanych reguł przejścia
d) Przycisk uruchamiający rysowanie
e) Płótno, na którym wyświetlane są wyniki; komórki żywe reprezentują kwadraty koloru czerwonego. Rozmiar komórek dostosowany jest do rozmiaru przestrzeni i liczby iteracji, największy możliwy rozmiar siatki to 600x600

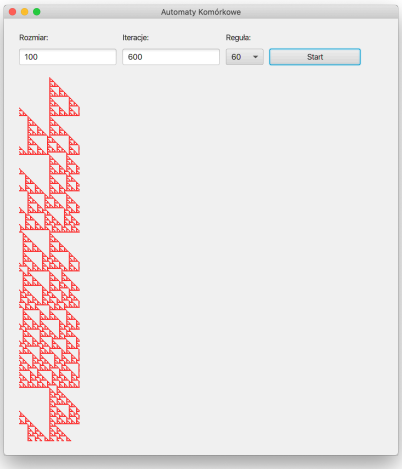
2. Wyniki

Wyniki działania aplikacji przedstawiają wydruki na rysunkach 2.- 6. Kolejne iteracje są ukazane jako ułożone jeden pod drugim rzędy kwadratów dwóch kolorów. Użyto parametrów:

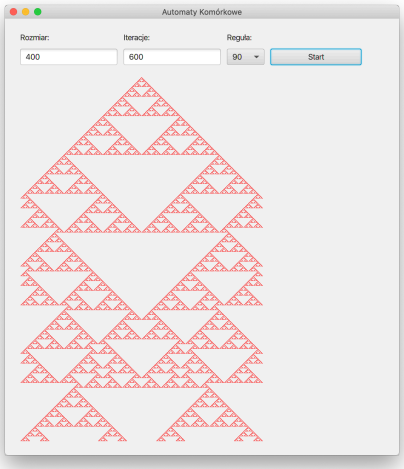
a) Reguła przejścia: 30, rozmiar przestrzeni: 100, liczba iteracji: 150, przedstawiony na rys. 2
b) Reguła przejścia: 60, rozmiar przestrzeni: 100, liczba iteracji: 600, przedstawiony na rys. 3
c) Reguła przejścia: 90, rozmiar przestrzeni: 400, liczba iteracji: 600, przedstawiony na rys. 4
d) Reguła przejścia: 120, rozmiar przestrzeni: 50, liczba iteracji: 600, przedstawiony na rys. 5
e) Reguła przejścia: 225, rozmiar przestrzeni: 600, liczba iteracji: 600, przedstawiony na rys. 6



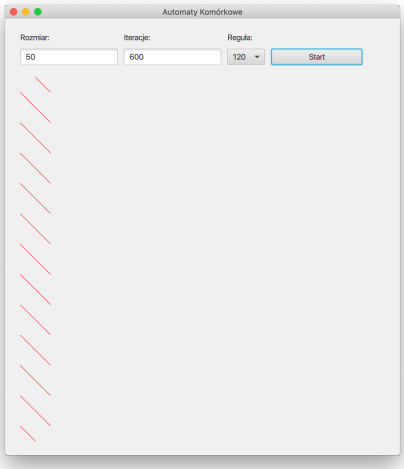
Rys. 2 Wynik działania aplikacji dla reguły 30



Rys. 3 Wynik działania aplikacji dla reguły 60



Rys. 4 Wynik działania aplikacji dla reguły 90



Rys. 5 Wynik działania aplikacji dla reguły 120

Rys. 6 Wynik działania aplikacji dla reguły 225

3. Spełnione wymagania

a) Stan początkowy automatu to jedna żywa komórka na środku przestrzeni
b) Zaimplementowano automaty z regułami przejścia: 30, 60, 90, 120, 225 (każda reguła jest zaprezentowana na jednym rysunku w punkcie 2.)
c) Zastosowano periodyczny warunek brzegowy (najlepiej potwierdza to rysunek 5.)
d) Wyniki są przedstawione w formie graficznej (punkt 2.)
e) Zmiany parametrów użytkownik może dokonać w GUI (przedstawiono na rys. 1.)

4. Kod

Komórkę reprezentuje zmienna typu *boolean*, żywym komórkom przypisana jest wartość *true*, martwym - *false.* Automat jest reprezentowany w kodzie dwuwymiarową tablicą, której każdy wiersz odpowiada stanowi przestrzeni w jednej iteracji. Taka reprezentacja umożliwia narysowanie wszystkich iteracji jednym wywołaniem funkcji *draw.*

Podstawą prawidłowego działania automatu jest określenie sąsiedztwa danej komórki. Kod, który znajduje kombinację sąsiadów zadanego elementu tablicy pokazuje Listing 1. Kombinacje są reprezentowane liczbami, które przedstawiają w systemie binarnym, zapisanymi w systemie dziesiętnym.

Kolejnym etapem jest zastosowanie reguły przejścia (na Listingu 2. przedstawiono stosowanie reguły 30). Wybór odpowiedniej regułę realizowany jest z użyciem instrukcji switch case. Dla każdego elementu przestrzeni wyznaczana jest kombinacja sąsiadów z poprzedniej iteracja, i w zależności od numeru kombinacji i zastosowanej reguły komórka jest określana jako martwa lub żywa.

```
private int getNeighbourhood(
        boolean[][]grid, int i, int j, int size){
    boolean a, b, c;
    if(j==0) a=grid[i-1][size-1];
    else a=grid[i-1][j-1];
    b=grid[i-1][j];
    if(j==size-1) c=grid[i-1][0];
    else c=grid[i-1][j+1];
    if(a) {
        if (b) {
            if (c) return 7;
            else return 6; }
        else if(c) return 5;
        else return 4; }
    else if(b) {
        if (c) return 3;
        else return 2; }
    else if(c) return 1;
    else return 0; }
```

Listing 1. Określanie sąsiedztwa

```
for(int i =1; i< iteration; i++){
    for (int j = 0; j < size; j++) {
        neighbourhood= getNeighbourhood(gd, i, j, size);
        if (neighbourhood==1||neighbourhood==2||
                neighbourhood==3||neighbourhood==4)
            gd[i][j]=true;
        else
            gd[i][j]=false; } }
break;
```

Listing 2. Stosowanie reguły przejścia