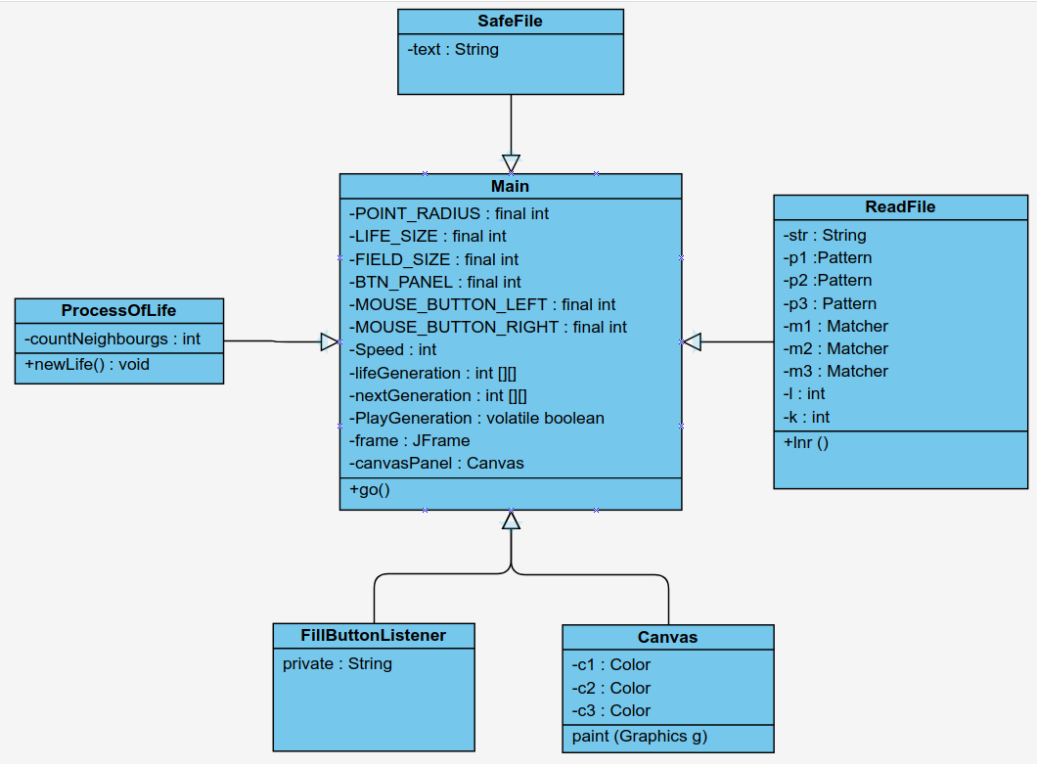
SPECYFIKACJA IMPLEMENTACYJNA "WireWorld" DLA JĘZYKA PROGRAMOWANIA JAVA

Spis treści

1 Diagram klas



Rysunek 1: Diagram klas

1.1 Opis diagramu

Poniżej jest przedstawiony opis diagramu:

`Main` jest główną klasę. Klasa `Main` wiąże między sobą wszystkie poszczególne klasy w programie "WireWorld".

Metoda `void go()` zawiera cały interfejs graficzny oraz zarządzanie kliknięciem myszki i przesunięciem suwaka i odpowiada za `Start,Stop,NewGame`. Metoda `void go()` odpowiada za prędkość przesunięcia komórek.

W klasie `ReadFile` mamy `lnr` - LineNumberReader, który wczytuje poszczególne wartości z pliku `.txt`.

W klasie `Canvas` mamy metodę `public void paint(Graphics g)`, za pomocą której robimy graficzny interfejs dla pola i komórek.

2 Opis metod/pakietów

Cały program został napisane w jednej klasie `Main.java`, bo na czas dzisiejszy innego rozwiązania problemu nie znaleźliśmy.

Kod programy jest napisany za pomocą dwóch tablic dwuwymiarowych i zmiennej typu `boolean` dla uruchomienia (Play) lub dla zatrzymania (Stop):

- `int[][] lifeGeneration`,
- `int[][] nextGeneration`,
- `volatile boolean PlayGeneration`.

2.1 Klasa Main

Klasa `Main` zawiera w sobie klasy i następujące pola stałe:

- `final int POINT_RADIUS` odpowiada za punkt radiusa,
- `final int LIFE_SIZE` odpowiada za lewy rozmiar punktu,
- `final int FIELD_SIZE` odpowiada za rozmiar okienka,
- `final int BTN_PANEL` odpowiada za rozmiar paneli przycisków,
- `final int MOUSE_BUTTON_LEFT` odpowiada za lewy przycisk,
- `final int MOUSE_BUTTON_RIGHT` odpowiada za prawy przycisk,
- `int Speed` odpowiada za suwak prędkości, wartość maksymalnej prędkości.

2.2 Metoda go()

Metoda `void go()` zawiera w sobie GUI i kod działania programu, które napisane poprzez bibliotekę **Swing**, jest stworzony `JFrame WireWorld` z odpowiednimi rozmiarami i wyglądem. Tak samo stworzony `JButton Play` dla uruchomienia działania programu.

Za pomocą `canvasPanel.addMouseListener` wyznaczamy naciski na myszkę, to znaczy, uzupełniamy wartościami:

- `MOUSE_BUTTON_LEFT`,
- `MOUSE_BUTTON_RIGHT`.

W tej metodzie wyznaczamy suwak prędkości zależne od położenia wciśniętej myszki, za pomocą `public void stateChanged(ChangeEvent e)`.

2.3 Metoda processOfLife

Metoda `void processOfLife()` liczy sąsiedztwa metodą Moore'a za pomocą dwóch dwuwymiarowych tablic. Tablica `lifeGeneration` odpowiada za stan teraźniejszy pozycji punktu, a tablica `nextGeneration` odpowiada za następującą generację punktów.

2.4 Metoda newLife

Metoda `void newLife()` oczyszcza całe plansze od wszystkich punktów i robi `newGame`.

2.5 Metoda saveFile

Metoda `void saveFile()` zapisuję wszystkie wygenerowane punkty do pliku w postaci:

```
Field: 18, 20;
ElectronTail: 19, 20;
Field: 20, 20;
ElectronHead: 21, 19;
Field: 21, 20;
```

Gdzie `Field` to koordynata punktu, `ElectronTail` to ogon elektronu, `ElectronHead` to głowa elektronu. Z kolej wynika, że pole `Field` zawiera w sobie `Diodę`.

2.6 Metoda readFile

Metoda `void readFile()` czyta punktu z pliku w postaci:

```
Field: 18, 20;
ElectronTail: 19, 20;
Field: 20, 20;
ElectronHead: 21, 19;
Field: 21, 20;
```

Gdzie `Field` to koordynata punktu, `ElectronTail` to ogon elektronu, `ElectronHead` to głowa elektronu. Z kolej wynika, że pole `Field` zawiera w sobie `Diodę`.

2.7 Pakiet

Cały program "WireWorld" jest napisany w `package gra`, bo zdecydowaliśmy, że w ten sposób będzie łatwej.

3 Metodyka "WireWorld"

3.1 Zasady

Program "WireWorld" wykorzystuje zestaw zasad:

- Komórka pozostaje Pusta, jeśli była Pusta,
- Komórka staje się Ogonem elektronu, jeśli była Głową elektronu,
- Komórka staje się Przewodnikiem, jeśli była Ogonem elektronu
- Komórka staje się Głową elektronu tylko wtedy, gdy dokładnie 1 lub 2 sąsiadujące komórki są Głowami Elektronu,
- Komórka staje się Przewodnikiem w każdym innym wypadku.

3.2 Metoda sąsiedztw

Program napisany za pomocą metody sąsiedztw Moore'a. W sąsiedztwie Moore'a mamy 8 przylegających komórek (znajdujących się: na południu, na południowym-zachodzie, na zachodzie, na północnym-zachodzie, na północy, na północnym-wschodzie, na wschodzie i na południowym-wschodzie).

4 Opis GUI

Jest trochę opisany w metodzie **void go()**, ale tutaj bardziej szczegółowo:

- `JFrame` tworzy okienko o rozmiarze `500x570`,
- `JFrame.EXIT_ON_CLOSE` odpowiada za wyłączenia programu przy naciśnięciu na czerwony suwak krzyż,
- `Canvas` tworzy panel do wykorzystywanie i wciśnięcia myszką, żeby pojawiały się diodę, punktu,
- `JSlider` tworzy liniowe regulatory (suwaki), które dają możliwość wyboru konkretnej wartości prędkości z zakresu od 1 do 700,
- `JButton Step` pokazuję generacje krok po kroku,
- `JButton Play` symuluje generacje z prędkością 350,
- `JButton Stop` zatrzyma działania kolejnych generacji,
- `JButton NewGame` wyczyszcza całe pole i zaczyna program od nowa,
- `JButton Save` zapisuje generacje do pliku z rozszerzeniem `.txt`,
- `JButton Upload` otworzy plik `.txt` z odpowiednimi generacjami.

5 Testowanie

5.1 Użyte narzędzia

5.1.1 AssertJ

Testy jednostkowe będą robione z wykorzystaniem biblioteki **AssertJ**, która pozwala na nagrywanie zatwierdzenia w Java-testów.

5.1.2 Maven

Maven — narzędzie automatyzujące budowę oprogramowania na platformę Java. Plik określający sposób budowy aplikacji nosi nazwę POM-u (ang. Project Object Model). W naszym programie będziemy wykorzystywać **Mavena** aby zapewnić przenośność kodu na poziomie deweloperskim. Poprzez użycie **Mavena** powstanie plik wynikowy z rozszerzeniem `.jar`.