SPIS TREŚCI

1 Wprowadzenie 2
1.1 Czym jest automat komórkowy? 2
1.2 Mrówka Langtona jako automat komórkowy 2
2 Założenia 3
3 Rozwiązania architektoniczne 3
4 Szczegóły techniczne 3
5 Hierarchia pakietów 4

Mrówka Langtona(Langton's Ant)
wydanie na standardowej licencji :D

Wprowadzenie

Czym jest automat komórkowy?

Automat komórkowy – system składający się z pojedynczych komórek, znajdujących się obok siebie. Ich układ przypomina szachownicę lub planszę do gry. Każda z komórek może przyjąć jeden ze stanów, przy czym liczba stanów jest skończona, ale dowolnie duża. Stan komórki zmieniany jest synchronicznie zgodnie z regułami mówiącymi, w jaki sposób nowy stan komórki zależy od jej obecnego stanu i stanu jej sąsiadów. składający

Wikipedia

Mrówka Langtona jako automat komórkowy

W najprostszym przypadku: w każdym kroku wyróżnione są komórki nazywane "mrówkami". Mrówki zmieniają kolor swojej komórki i przechodzą do jednego z sąsiednich pól według określonych zasad.Gdy wejdzie na czarną komórkę zmienia jej pole na białe i skręca w lewo, natomiast w przypadku białej komórki zmienia na czarne i skręca w prawo (mrówka ma kierunek, który określa jakie pole w kolejnym ruchu odwiedzi)

W naszym projekcie będą możliwe dwie opcje symulacji mrówki.

W pierwszym przypadku będzie to symulacja dowolnie wybranej liczby mrówek, które będą miały własny kolor. Według poprzedniej definicji, mrówki będą rozpoznawały każdą "niebiałą" komórkę jako "czarną".

W drugiej symulacji będzie dostępna jedna mrówka, która będzie "malowała" w kilku kolorach. Natrafi na kolor numer 1, zamaluje na kolor numer 2, i tak dalej, aż do końcowego koloru - wtedy zamaluje na biało. Użytkownik nie będzie miał mozliwości wyboru kolorów, jednak określi ile ich będzie, oraz może określić zachowanie (określenie kierunku) dla każdego koloru oddzielnie.

Wstępne okienko pozwala określić ustawienia początkowe w tym: zachowanie mrówki, ile ruchów ma być wyświetlane, wielkość planszy po której mrówki się poruszają, maksymalna ilość kroków dla mrówki. W przypadku współrzędnych niepoprawnych, wykraczających poza granice planszy nie zostanie wyświetlony komunikat - po prostu takiej mrówki nie będzie na planszy.

Założenia

Założeniami w projekcie było aby:
•aplikacja posiadała szeroki zakres funkcji,
•była odporna na zły input,
•można było wczytywać zapisane wcześniej płaszczyzny,
•była wygodna w użyciu ,
•dostosowywała się do rozmiarów ekranu użytkownika
•aplikacja była wszestronna - możliwość zarówno wczytywania dowolnej ilości mrówek dla wersji podstawowej, jak i możliwość reagowania jednej mrówki na wiele kolorów.

Rozwiązania architektoniczne

Żeby umożliwić łatwy zapis i wczytywanie stworzono serializowalne wrappery (adaptery) dla płaszczyzny, oraz klasy Color.
Dla zmniejszenia repetytowności kodu oraz ujednolicenia interfejsu klasa abstrakcyjna AbstractAntCore, po której dziedziczą oba pozostałe.
Dla zwiększenia czytelności kodu interfejs graficzny napisany w FXMLu oraz backend tego interfejsu w określonych dla danego FXMLa kontrolerach
Za odtwarzanie animacji odpowiedzialna jest klasa javafx.animation.Timeline, która pracuje na wątku aplikacji JavaFX, dzięki czemu uniknęliśmy problemów przy pracy z równoległymi wątkami - wyjątki, freezowanie aplikacji.
Klasa ConfigurationSetup umożliwia łatwe przeniesienie potrzebnej konfiguracji do kontrolera okna animacji, dzięki czemu została uniknięta przytłaczająco duża ilość zmiennych/instancji statycznych.
Pomimo że kolory są specyfikowane w kodzie, przy każdym pojawieniu się okna konfiguracji są one wymieszane, co powoduje inną animację za każdym razem, zamiast poczucia z góry ustalonego schematu.

Szczegóły techniczne

Projekt został przygotowany w Javie 8, w środowisku Intellij IDEA (https://www.jetbrains.com/idea/), do przygotowania graficznego interfejsu skorzystaliśmy z JavaFx.

Hierarchia pakietów

Diagram zależności pakietów