SPIS TREŚCI

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.

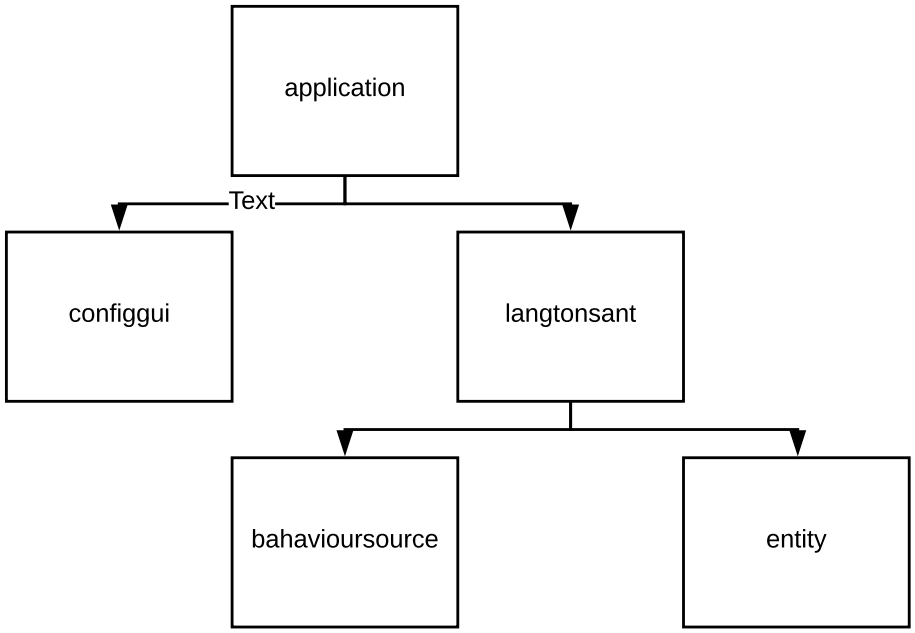
Wstępne okienko pozwala określić ustawienia początkowe w tym: zachowanie mrówki, ile ruchów ma być wyświetlane, wielkość planszy po której mrówki się poruszają, maksymalna ilość kroków dla mrówki. W przypadku współrzędnych niepoprawnych, wykraczających poza granice planszy nie zostanie wyświetlony komunikat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

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



Klasy w projekcie

Main

Klasa dostarcza interfejs graficzny.
metody:
•private void displayConfigDialog() throws IOException
inicjuje okno interfejsu graficznego, gdzie wybiera się ustawienia mrówek. Metoda korzysta z configurationGui.fxml dla szczegółów okienka
•private void setupAnimationWindow(Stage primaryStage)
ustawia elementy wewnątrz okna, w tym przyciski i ich obslugę. Daje możliwość zapauzowania, zapisu, oraz możliwość cofnięcia
•private void resetApplication()
Metoda z której korzysta przycisk backToSettings. Ładuje na nowo okienko do ustawienia mrówek
•private void saveCurrentAntCore(SavableAntCore savableAntCore, File file) Metoda ma za zadanie zapisac klasy potrzebne do późniejszego wczytania animacji.

pola:
public static int refreshDelay
-wartość opóźnienia animacji (fps)
public static Long stepsLimit
-maksymalna ilość kroków mrówki
public static Long currentSteps
public static List<Ant> antList
-kontener dla mrówek public static Map<Integer, SavableColor> colorMap
-Mapa która każdej wartości(id mrówki) przyporządkowuje unikalny kolor
public static Canvas canvas
public static GraphicsContext graphicsContext
public static Controller controller
public static Plane plane

private Timeline timeline
private SavableAntCore currentAntCore

pakiet behaviourcore

BasicAntCore

Klasa ta zajmuje się obsługą wielu mrówek
metody:
•private void antSetDirection(Ant ant)
-metoda ustawia kierunek mrówki, zmienia aktualny kierunek w zależności od koloru na jakie trafił
•private void antSetPlaneValue(Ant ant)
-metoda wprowadza zmiany na tablicę wartości, ustawia nową pozycję mrówki
•private void antStep(Ant ant)
-metoda pakuje metody odpowiedzialne za krok mrówki
•private boolean iterateThroughAntList()
-metoda odpowiada za wszystkie mrówki. Monitoruje aktywność, sprawdza, czy zderzyly się ze ścianą, jeśli nie pozwala wykonać ruch.

pola:
private List<Ant> antList

CustomAntCore

Klasa ta zajmuje się obsługą mrówki kolorowej
metody:
•public CustomAntCore(Plane plane, Ant theOnlyAntThatMatters, Map<Integer, SavableColor> colorMap,GraphicsContext graphicsContext, int antRectangleSize)
Klasa odpowiada za obsługę mrówki z wieloma kolorami, interpretuje podany ciąg(set behaviour String) jako zachowanie mrówki
•private void markAsVisited(Ant ant)

•private void antSetDirection(Ant ant)
-ustawia kierunek biały->prawo, niebialy->lewo
•private void antSetPlaneValue(Ant ant)
-wprowadza aktualne położenie mrówki na tablicę wartości
•private void antStep (Ant ant)

-robi krok dla mrówki kolorowej, pakuje metody odpowiedzialne za jeden cykl

pola:
private Ant theOnlyAntThatMatters
private int behaviourLimit

SavableAntCore

Klasa umożliwia zapis Klasy do strumienia (imeplementuje Serializable)
metody:
•protected void preFillPlane()
-maluje w okienku obraz w oparciu o tablicę wartości
•protected void fillPlaneOnPosition(int x, int y)
-sprawdza komórkę o podanych współrzędnych w tablicy wartości i maluje odpowiedni kolor komórki na ekranie

pola:
•protected int antRectangleSize
•protected Plane plane
•protected GraphicsContext graphicsContext
•protected Map<Integer, SavableColor> colors

pakiet entity

Ant

Klasa przechowuje parametry dla pojedynczej mrowki
metody:
•public void antMove()
-wykonuje krok dla jednej mrówki
•public boolean checkIfCrashed(int planeSize)
-sprawdza, czy mrówka nie wyszła poza tablicę, jeśli tak, nie będzie już wykonywała ruchu
•public void interpretBehaviourString (String behaviourString)
-metoda sprawdza wprowadzone zachowanie dla mrówki - dzięki niej możliwe jest ustawienie zachowania dla każdego koloru.
•public void setStartDirection()
-ustawia kierunek mrówki
•public int getX()
-metoda dostępowa współrzędnej X
•public void setX(int x)
-metoda dostępowa współrzędnej X
•public int getY()
-metoda dostępowa współrzędnej Y
•public void setY(int y)
-metoda dostępowa współrzędnej Y
•public int getDirection()
-metoda dostępowa parametru direction
•public void setDirection(int direction)
-metoda dostępowa parametru direction
•public boolean isActive()
-metoda dostępowa parametru isActive
•public void setActive(boolean active)
-metoda dostępowa parametru isActive
•public int getId()
-metoda dostępowa parametru id

pola:
•private long serialVersionUID = 1L
•private static int antNumber
-statyczna zmienna do liczenia mrówek

•private final int id
-identyfikator który jest wprowadzany na tablicę wartości
•private int x
-współrzędna X
•private int y
-współrzędna Y
•private int direction
-kierunek: 0->góra, 1->prawo, 2->dół, 3->lewo
•private boolean isActive
-wartość mówi, czy mrówka jeszcze jest aktywna, czy się jeszcze nie zderzyła
•public Direction[] behaviourArray
-jest to zinterpretowane zachowanie dla każdego koloru pojedynczo w przypadku mrówki wielokolorowej

Plane

Klasa przechowuje parametry dla tablicy wartości, po której chodzą mrówki
metody:
•public void printPlane()
-wypisuje na konsole tablice wartości
•public int[][] getPlane()
-zwraca tablicę wartości
•private void setPlane(int[][] plane)
-ustawia tablicę wartości
•public int getValueOnPosition(int x, int y)
-zwraca wartość dla podanej współrzędnej
•public void setValueOnPosition(int x, int y, int value)
-ustawia wartość dla podanej współrzędnej
•public int getPlaneSize()
-zwraca wielkość tablicy (bok kwadratu)

pola:
•long serialVersionUID = 1L
•private int[][][] plane
•private int planeSize

SavableColor

Klasa pomocnicza przy wczytywaniu (zapisie) mrówki
metody:
•public void setFromColorClass(Color color)
•public Color toColorClass()

pola:
•private long serialVersionUID = 1L
•private double r
•private double g
•private double b
•private double opacity