Uniwersytet Gdański
Informatyka
1 st. III semestr
Dawid Bińkuś

Anthill (Mrowisko)

– Dokumentacja projektu –

Gdańsk, 07.11.2017

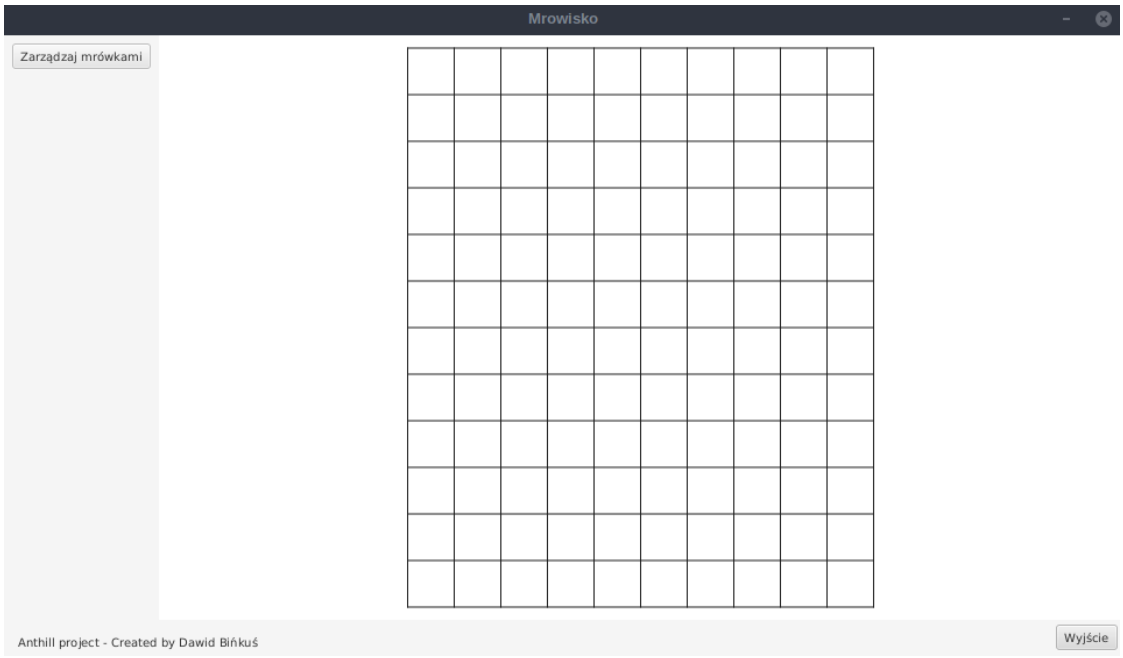
Table of Contents

1 Wstępne informacje dotyczące projektu

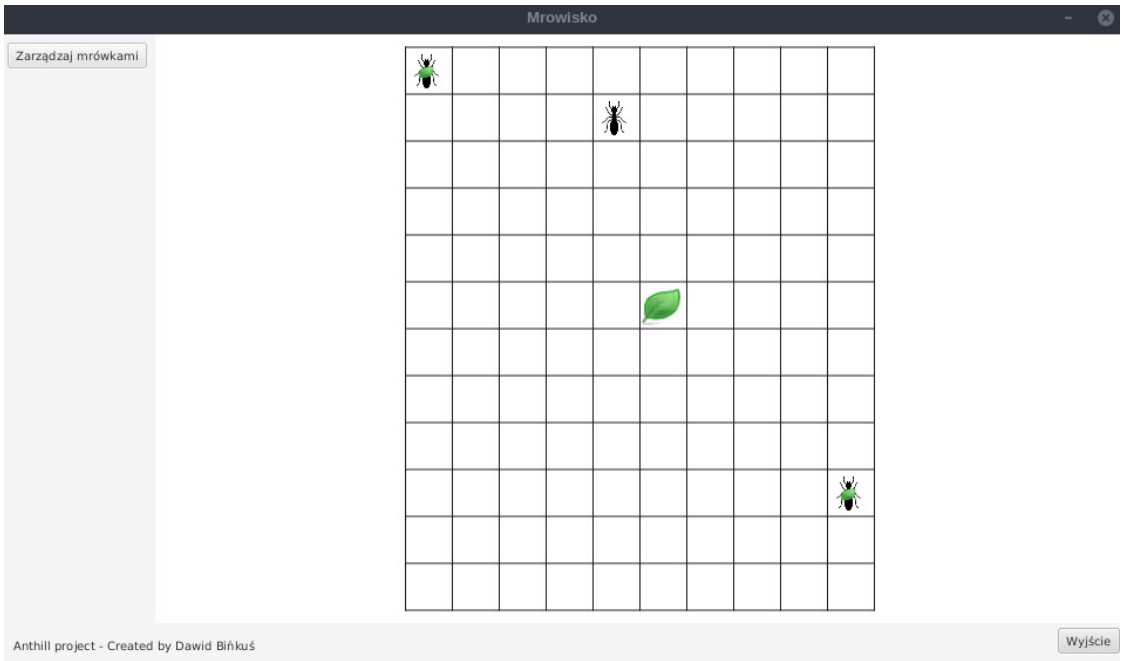
Dokumentacja dotyczy symulatora mrowiska – programu graficznego, który umożliwia użytkownikowi manipulację stanem mrówek i ich pozycją w mrowisku. Każda nowoutworzona mrówka ma przypisane imię oraz swoje koordynaty na siatce mrowiska. Dodatkowo mrówki mają możliwość podnoszenia liści, których stan również zależy od użytkownika. Pole mrowiska przedstawiane jest jako siatka A x B (na chwilę obecną narzucaną z góry przez programistę) a współrzędne X i Y są rozumiane kolejno jako oś pozioma oraz oś pionowa mrowiska (lewy górny punkt ma współrzędne (0,0), dolny prawy zaś (A-1,B-1)).

Oprócz zarządzania stanem mrówek i liści w mrowisku, użytkownik ma również możliwość poruszania wybraną mrówką za pomocą klawiszy WSAD po uprzednim wybraniu mrówki z listy. Gdy mrówka nieposiadająca listka wejdzie na pole, na którym taki liść się znajduje, zostaje on automatycznie podniesiony przed daną mrówkę i przypisany do niej. Od tej pory nosi go na swoich plecach. Gdy taka mrówka ponownie wejdzie na pole z liściem, wyrzuci ona liść na swoich plecach na pobliskie pole. To samo uczyni z liściem, na który miała zamiar wejść, by pole, które miała zamiar zająć było puste.

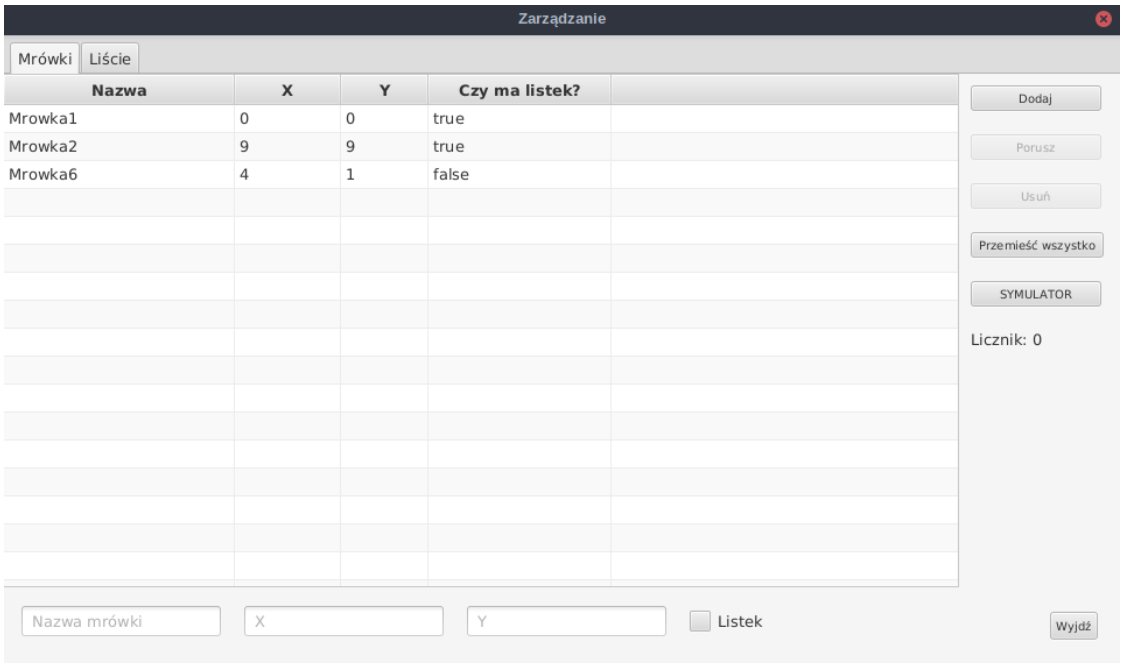
Oprócz manualnego sterowania jedną mrówką, użytkownik może poruszyć wszystkimi mrówkami w tym samym czasie za pomocą specjalnej opcji (liście nie zmieniają swojej pozycji). Dodatkowo została dostarczona opcja symulacji pracy mrowiska - mrówki zmieniają wtedy swoje położenie w czasie rzeczywistym do momentu wyłączenia przełącznika przez użytkownika. Podczas uruchomionej symulacji użytkownik wciąż może manipulować stanem mrówek.



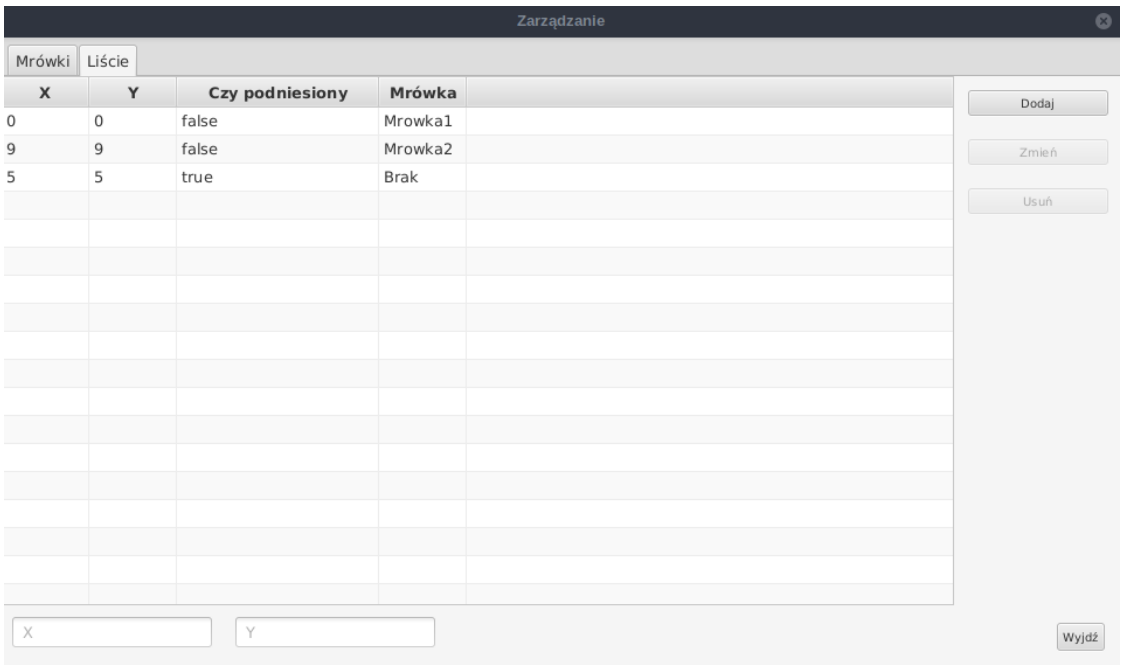
Okno główne programu



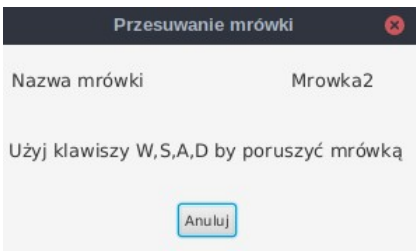
Okno główne programu z elementami na siatce



Okno zarządzania mrówkami



Okno zarządzania liśćmi



Okno poruszania mrówką

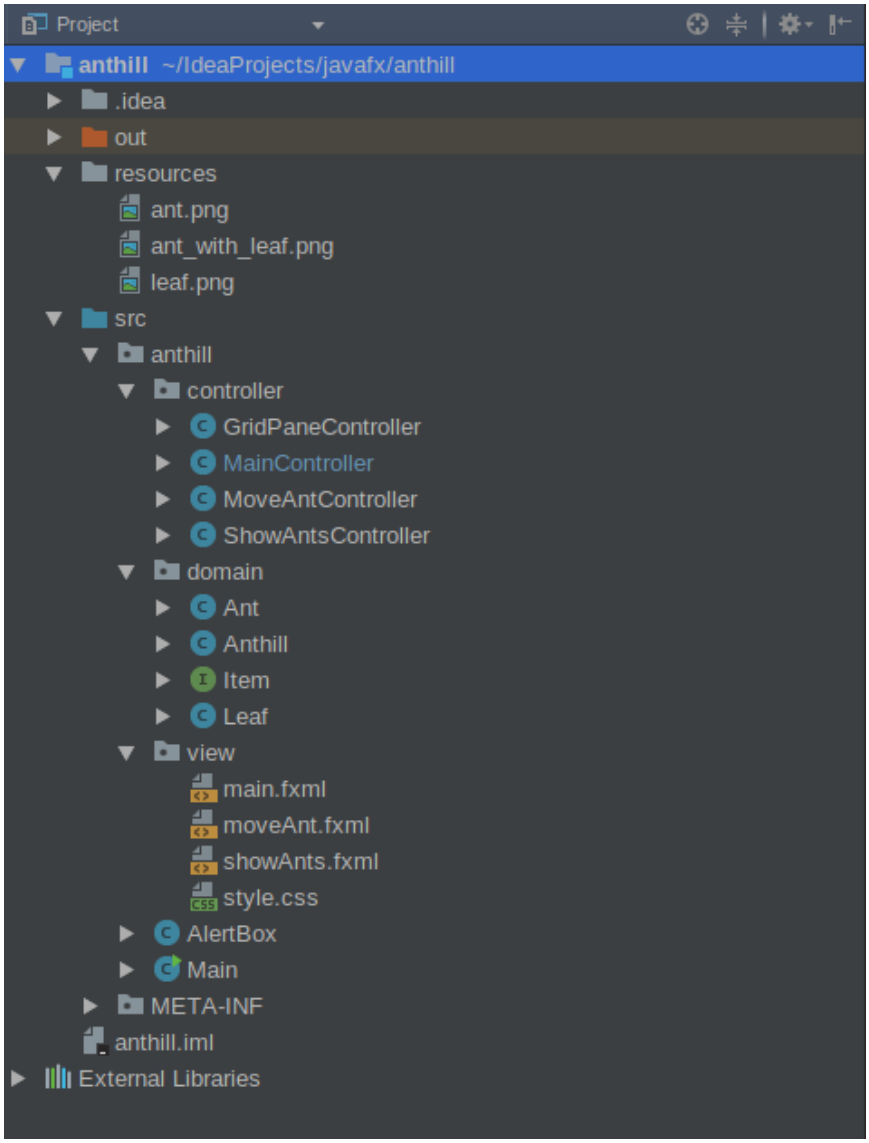
2 Organizacja projektu

2.1 Użyte technologie

Projekt został napisany z użyciem języka Java w wersji 1.9. Za graficzną stronę projektu odpowiada biblioteka wbudowana Java-FX. Użyte zostało również oprogramowanie od Oracle – Scene Builder, który znacząco ułatwił pracę z plikami FXML.

2.2 Podział projektu

Projekt został napisany z użyciem wzorca MVC(Model-View-Controller), którego zadaniem jest oddzielić logikę i widok aplikacji.
Organizacja projektu prezentuje się następująco:



Podział plików w projekcie Anthill

W kontekście analizy aplikacji interesują nas tylko foldery resources – zawierające pliki graficzne używane w programie oraz folder src (od angielskiego source) zawierający kod źródłowy aplikacji.
W katalogu src umieszczony jest pakiet anthill (będący pakietem korzeniem, zawierającym wszystkie inne) oraz jego trzy gałęzie:

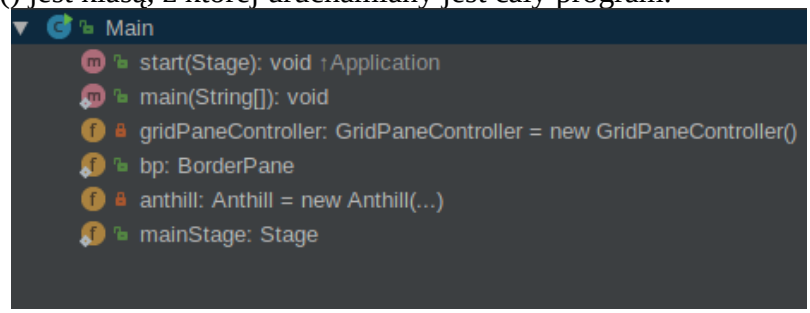
- view – zawierający pliki fxml odpowiadające za rozmieszczenie poszczególnych elementów w interfejsie użytkownika oraz plik stylów, określający ich wygląd
- domain(model) – zawierający klasy reprezentujące obiekty (mrówka i liść) oraz samo mrowisko
- controller – zawierający kontrolery, będące swoistym mostem, pomiędzy plikami w pakiecie view, oraz tymi w pakiecie domain

Oprócz tego, dodane są jeszcze dwa dodatkowe pliki, nie znajdujące się w żadnych podpakietach – Main i AlertBox. Wspomnę o nich w następnej części dokumentacji.

3 Analiza poszczególnych klas

3.1 Klasa Main

Klasa Main() jest klasą, z której uruchamiany jest cały program.



Jako, iż aplikacja jest napisana za pomocą biblioteki Java-FX, klasa Main musi rozszerzać klasę Application, która odpowiedzialna za wczytanie graficznego interfejsu użytkownika (w tym przypadku ładowany jest plik main.fxml). Wszystkie te operacje wykonywane są w metodzie start. Metoda main ma za zadanie tylko uruchomić aplikację za pomocą stworzonego wcześniej interfejsu w metodzie start.

Pola gridPaneController oraz bp są używane do dynamicznego generowania siatki mrowiska (całą logikę z tym związaną wykonuje gridPaneController o czym wspomnę później, bp zaś przechowuje układ głównego okna programu – występuje ono tutaj jako pole statyczne, by inne klasy miały do niego dostęp bez potrzeby tworzenia obiektu klasy Main)

Pole anthill słuzy do wywołania obiektu typu Anthill, wraz z podaniem takich informacji jak rozmiar mrowiska (w tym wypadku narzucone przez programistę).
Ostatnie już pole mainStage, służy do przekazania go innym klasom (m.in w celu manipulacji na obecnym oknie – np. jego zamknięciu)

3.2 Pakiet view

W pakiecie view znajdują się cztery pliki. Przeanalizujmy teraz każdy z nich:

3.2.1 Plik main.fxml

Plik main.fxml określa rozmieszczenie elementów na głównym oknie programu.
Jest on zbudowany na układzie nazywanym BorderPane (po więcej informacji tu: https://docs.oracle.com/javafx/2/api/javafx/scene/layout/BorderPane.html).
W pliku fxml określona jest tylko lewa część układu (zajmuje ją guzik służący do zarządzania mrowiskiem) oraz dolna (znajdują się tam informacje o autorze oraz przycisk służący do wyjścia z programu)

3.2.2 Plik showAnts.fxml

Plix showAnts.fxml określa rozmieszczenie elementów na oknie zarządzania mrowiskiem. Jest on zbudowany na układzie nazywanym TabPane (https://docs.oracle.com/javafx/2/api/javafx/scene/control/TabPane.html).
Zastosowanie tego układu umożliwiło rozdzielenie okien zarządzania mrowiskiem na dwie zakładki(mrówki i liście). Oba podokna są również typu BorderPane i wyglądają w bliźniaczy sposób – na dolnej i prawej części znajdują się przyciski i pola do wpisania danych przez użytkownika. W centralnej zaś znajduje się obiekt typu TableView, który przechowuje bazę kolejno mrówek i liści.

3.2.3 Plik moveAnt.fxml

Plik moveAnt.fxml jest prostym oknem, pojawiającym się gdy użytkownik wyrazi chęć poruszenia daną mrówką samodzielnie.
Zbudowany on jest na układzie nazywanym GridPane
(https://docs.oracle.com/javafx/2/api/javafx/scene/layout/GridPane.html)

Jest to proste okno, wyświetlające użytkownikowi imię wybranej mrówki, instrukcje dotyczące sterowania oraz przycisk wyjścia z trybu sterowania.

3.2.4 Plik style.css

Plik stylów zawiera informacje dotyczące kosmetyki interfejsu. Obecnie jest on na etapie wdrażania (umieszczone są tylko podstawowe informacje) ale umożliwiają one zmianę całego interfejsu w dosłownie kilka chwil.

3.3 Pakiet model

Pakiet model(domain) zawiera 3 klasy i 1 interfejs. Przyjrzyjmy się im teraz.

3.3.1 Interfejs Item

Interfejs Item ma duże znaczenie w przypadku operacji na obiektach znajdujących się w mrowisku. Jako sam plik nie zawiera zbyt wiele – tylko definicję kilku metod pobierających (getX igetY) i ustawiających(setX i setY) koordynaty danego obiektu bez ich implementacji.

Jednakże, interfejsy mogą być implementowane przez pewne klasy (w tym przypadku Ant i Leaf), dzięki czemu w niektórych metodach używanych później można odnosić się do obu tych obiektów za pomocą jednego typu pierwotnego (w tym wypadku Ant).

Więcej na ten temat znajduje się tutaj:
https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html

3.3.2 Klasa Ant

Klasa ant zawiera obiekty zawierające informacje o położeniu (x,y typu SimpleIntegerProperty, używanego w generowaniu wymienionego wyżej obiektu TableView), imieniu (analogicznie, tylko typu String), informacji czy mrówka nosi liść, przypisany konkretny liść(jeśli nie ma żadnego, pole ma wartość null) oraz obiekt typu ImageView (odpowiedzialny za przechowywanie aktualnego grafiki mrówki)

Z wyjątkiem jednej, metody zawarte w klasie są metodami dostępowymi (umożliwiają programiście ustawienie odpowiednich wartości, jeśli zajdzie taka potrzeba).
Metoda updateImage(), wywoływana w sytuacji, gdy tworzona jest nowa mrówka, bądź istniejąca mrówka wpada na liść, ma za zadanie zaktualizowanie grafiki.

3.3.3 Klasa Leaf

Klasa Leaf ma praktycznie identyczną strukturę co klasa Ant, z tą różnicą że zawiera informacje o przypisanej mrówce, i o tym, czy listek jest możliwy do podniesienia.

Metoda updateImage uwzględnia test, czy liść nie został utworzony na istniejącej mrówce by zaimplementować konkretne zachowania liscia (automatyczne przypisanie nowoutworzonego liścia do mrówki, jeśli koordynaty się pokrywają).

3.3.4 Klasa Anthill

Klasa Anthill zawiera informacje o wielkości mrowiska oraz obiekty typu ArrayList<T>, zawierające wszystkie mrówki i liście należące do danego mrowiska.

Metody zawarte w klasie Anthill służą wykonywaniu określonych operacji w mrowisku (losowe przejście mrówki), znajdowaniu konkretnych obiektów z mrowiska(bazując na koordynatach) oraz testom i logice(np. Zachowaniu obiektów po wstąpieniu na liść).

3.4 Pakiet controller

Pakiet controller zawiera 4 kontrolery:

3.4.1 GridPaneController

Kontroler ten jest odpowiedzialny za generowanie układu GridPane (informacje o tym układzie znajdowały się przy wspomnieniu o pakiecie view).
Układ ten, jest następnie wstrzykiwany do głównej sceny (w main.fxml) na pozycji Center. Skutkuje to wyświetleniem w głównym oknie aplikacji wygrenerowanego podglądu mrowiska (z zaktualizowaną pozycją obiektów).

3.4.2 MainController

Kontroler ten jest odpowiedzialny za reagowanie na akcje użytkownika na głównym ekranie aplikacji. Możemy wyróżnić dwie główne:

- showAntsButtonClicked() - uruchamia kolejne okno aplikacji z pliku showAnts.fxml. Dodatkowo upewnia się, że uzytkownik wyjdzie z okna, zanim zrobi coś w poprzednim (za pomocą funkcji initModality).
- exitButtonClicked() - wyłącza program, gdy użytkownik wciśnie przycisk

3.4.3 MoveAntController

Kontroler ten jest odpowiedzialny za reagowanie na akcje użytkownika podczas trybu poruszania mrówką. Możemy wyróżnić:

- moveAnt(KeyEvent) – oczekuje na wciśnięcie klawisza przez użytkownika. W sytuacji gdy jest to jeden z klawiszy W,S,A,D, mrówka wykonuje ruch w danym kierunku.
- CancelButtonClicked() - wyłącza okno trybu poruszania mrówką, gdy użytkownik wciśnie żądany przycisk

3.4.4 ShowAntsController

Kontroler ten jest odpowiedzialny za reagowanie na akcje użytkownika podczas trybu zarządzania mrówkami. Możemy wyróżnić:

- addButtonClicked() i addLeafButtonClicked() - pobierają dane z odpowiednich pól (tzw. TextField) i używają ich by stworzyć nowy obiekt
- deleteButtonClicked() i removeLeafButtonClicked() - usuwają zaznaczony obiekt (jeśli żaden nie zostanie wybrany, przycisk staje się “nieklikalny”)
- moveButtonClicked() - wywołuje okno moveAnt.fxml wraz z przypiętym do niego kontrolerem MoveAntController
- changeLeafButtonClicked() - pobiera dane z pól i zmienia koordynaty danego listka – jeśli żaden nie będzie wybrany, przycisku nie da się kliknąć. Wprowadzone zostało ograniczenie, które zabrania przenosić listków, znajdujących się na mrówkach.
- moveAllButtonClicked() - porusza wszystkimi mrówkami dokładnie jeden raz
- simulateAnthillToogleButtonClicked – przełącznik, który włączony uruchamia tryb symulacji, w którym mrówki poruszają się w dowolnym kierunku a użytkownik wciąż ma dostęp do modyfikacji (dzięki działaniu na innych rdzeniach). Pod przyciskiem znajduje się licznik wykonanych ruchów przez mrówki.
- exitButtonClicked() - wyłącza okno trybu zarządzania
- unlockButtons() - odpowiada za “nieklikalność” niektórych guzików w oknie
- reload() - odpowiada za odświeżenie podglądu mrowiska
- initialize(URL, ResourceBundle) – jako, iż kontroler ShowAntsController implementuje interfejs Initializable, ma on możliwość wykonania pewnej akcji

zaraz po uruchomieniu. Te tutaj służą do wstępnej konfiguracji obiektów TableView, by wyświetlić listę obiektów.

3.5 Klasa AlertBox

Klasa AlertBox jest klasą pomocniczą, ułatwiająca programiście pracę. Wywołanie obiektu tej klasy z odpowiednimi parametrami(nazwa okna oraz wiadomość) umożliwia stworzenie małego okienka z powiadomieniem że coś poszło nie tak(uniemożliwia też użytkownikowi pracę, dopóki nie wyłączy okna). Służy ono głównie do komunikacji z użytkownikiem.

4 Uruchamianie

Program można wykonać za pomocą pliku jar. Należy jednak pamiętać, by uzywać JRE w wersji 1.9 lub wyżej. Drugą opcją pozostaje pobranie kodu źródłowego i skompilowanie go samodzielnie

Link do repozytorium github: https://github.com/inql/anthill-fx-mvc