Programowanie obiektowe - projekt
Dokumentacja

Mikołaj Chmielecki, Jakub Mroziński

14.06.2020

Spis treści

1 Analiza czasownikowo - rzeczownikowa **2**

2 Karty CRC **2**

3 Diagram przypadków użycia **6**

4 Diagramy klas **7**

5 Diagramy obiektów **10**

6 Diagramy sekwencji **11**

7 Diagramy aktywności **14**

8 Diagramy maszyny stanów **18**

9 Link do repozytorium **20**

1 Analiza czasownikowo - rzeczownikowa

Symulacja łąki

Język: Java

Symulacja polega na pokazaniu łąki (o ustawionym przez użytkownika rozmiarze) po której poruszają się zwierzęta. Będą nimi: krowy, owce, wilki, koty i myszy. Na początku działania symulacji zostaną one rozmieszczone na losowych polach w losowej liczebności (jednak maksymalna i minimalna początkowa liczba osobników każdego gatunku będzie ustalona przed rozpoczęciem symulacji). Każdy gatunek zwierząt będzie się poruszał po łące z charakterystyczną prędkością i z czasem będzie się starzeć i umierać ze starości. Aby zapobiec wymarciu gatunku przy spotkaniu dwóch zwierząt przeciwnej płci powstanie trzecie o zerowym wieku i posiadające jedną z dwóch płci (50% prawdopodobieństwa).

Zwierzęta różnych gatunków przy spotkaniu na jednym polu będą mogły wchodzić ze sobą w interakcje. Koty po spotkaniu z myszami zjadają je. Wilki, w celu zyskania pożywienia, mogą zaatakować wszystkie zwierzęta, ale prawdopodobieństwo przeprowadzenia skutecznego ataku nie jest stuprocentowe: z myszą – 80%, z kotem – 60%, z owcą – 40%, z krową – 20%. Poza tymi przypadkami na jednym polu nie może przebywać więcej niż jedno zwierzę.

Na wolnych polach podczas działania symulacji będzie się pojawiało losowo rozmieszczane pożywienie potrzebne zwierzętom do przetrwania. Będzie to trawa (dla krów i owiec) oraz ser (dla myszy). Gdy jakieś zwierzę napotka na pożywienie którym nie może się pożywić to zostaje ono zniszczone. Przy krawędzi łąki będzie usytuowany wodopój (jeden lub więcej – ustala użytkownik przed rozpoczęciem symulacji), z którego będą mogły w każdej chwili korzystać aby zaspokoić pragnienie.

Symulacja zakończy się gdy któryś z gatunków osiągnie ustaloną przed rozpoczęciem symulacji liczebność lub gdy symulacja wykona określoną ilość iteracji. Po zakończeniu zostaną pokazane statystyki symulacji dla każdego gatunku: maksymalna liczba zwierząt, całkowita liczba zwierząt, liczba zabitych oraz czas trwania symulacji. Statystyki te zostaną również zapisane do pliku, którego nazwa zostanie podana na początku.

2 Karty CRC

Classname: Animal	
Superclass: none **Subclass(es):** Cat, Cow, Mouse, Sheep, Wolf	
Responsibilities: The class contains parameters and operations that can be performed on each animal.	**Collaboration:** Field, Meadow, Simulation

Classname: AnimalCreator	
Superclass: none **Subclass(es):** none	
Responsibilities: The class is responsible for creating a given number of animals and placing them randomly on the meadow.	**Collaboration:** Cat, Cow, Mouse, Sheep, Wolf, Field, Meadow

Classname: AnimalStats	
Superclass: none **Subclass(es):** none	
Responsibilities: The class is responsible for collecting statistics on animal populations during the simulation.	**Collaboration:** Cat, Cow, Mouse, Sheep, Wolf

Classname: Cat	
Superclass: Animal **Subclass(es):** none	
Responsibilities: The class stores the value of the speed at which cats move, and operations specific to this species. It makes it possible to distinguish between cats and other animals.	**Collaboration:** AnimalStats

Classname: Cow	
Superclass: Animal **Subclass(es):** none	
Responsibilities: The class stores the value of the speed at which cows move, and operations specific to this species. It makes it possible to distinguish between cows and other animals.	**Collaboration:** AnimalStats

Classname: Mouse	
Superclass: Animal **Subclass(es):** none	
Responsibilities: The class stores the value of the speed at which mouses move, and operations specific to this species. It makes it possible to distinguish between mouses and other animals.	**Collaboration:** AnimalStats

Classname: Sheep	
Superclass: Animal **Subclass(es):** none	
Responsibilities: The class stores the value of the speed at which sheeps move, and operations specific to this species. It makes it possible to distinguish between sheeps and other animals.	**Collaboration:** AnimalStats

Classname: Species	
Superclass: none **Subclass(es):** none	
Responsibilities: It makes it possible to get species names and perform methods for obtaining and clearing statistics for each of them.	**Collaboration:** Cat, Cow, Mouse, Sheep, Wolf, SaveAsCSV

Classname: Wolf	
Superclass: Animal **Subclass(es):** none	
Responsibilities: The class stores the value of the speed at which wolves move, and operations specific to this species. It makes it possible to distinguish between wolves and other animals.	**Collaboration:** AnimalStats

Classname: Feed	
Superclass: none **Subclass(es):** none	
Responsibilities: The class stores information on given type of food, i.e. current number, maximum number, number of eaten and number of destroyed	**Collaboration:** Field, SaveAsCSV

Classname: Field	
Superclass: none **Subclass(es):** Waterhole	
Responsibilities: The class stores information about the content of a given field in a meadow. It contains information on whether there is food for animals in a given field and stores this food. It also stores its position on meadow and list of animals that are on it.	**Collaboration:** Animal, Feed, Meadow

Classname: Meadow	
Superclass: none **Subclass(es):** none	
Responsibilities: The class stores meadow state information, consists of subfields, arranges waterholes and food during board initialization and arranges new food during the simulation.	**Collaboration:** Field, Simulation

Classname: Waterhole	
Superclass: Field **Subclass(es):** none	
Responsibilities: The class stores information on the number of waterholes in the meadow. It makes it possible to distinguish between a waterhole and a regular field.	**Collaboration:** none

Classname: Main	
Superclass: none **Subclass(es):** none	
Responsibilities: The class is responsible for starting and ending the simulation. Stores the path to the statistics output file. The class generates and stores statistics after the simulation.	**Collaboration:** Simulation, Parameters, StartFrame

Classname: Parameters	
Superclass: none **Subclass(es):** none	
Responsibilities: The class sets, stores and validates initial parameters, i.e. the minimum and maximum numbers of each animal species, the dimensions of the meadow and the number of waterholes.	**Collaboration:** SaveAsCSV, StringConverter

Classname: SaveAsCSV	
Superclass: none **Subclass(es):** none	
Responsibilities: The class is responsible for saving simulation parameters and statistics in a .csv file.	**Collaboration:** Species, Feed, Parameters

Classname: Simulation	
Superclass: none **Subclass(es):** none	
Responsibilities: The class is responsible for the simulation. In it there is the main simulation loop. It coordinates the actions of animals and forces interactions between them, such as reproduction, quenching thirst or hunger. Generates animals and gives the signal to the Meadow class to initialize. Displays the current state of the simulation. It is responsible for checking the end conditions of the simulation.	**Collaboration:** Meadow, Animal, Main

Classname: StringConverter	
Superclass: none **Subclass(es):** none	
Responsibilities: The class that supports the parser. Converts a string to a list.	**Collaboration:** none

Classname: AnimalStatsFrame	
Superclass: JFrame **Subclass(es):** none	
Responsibilities: The class is responsible for displaying the statistics window for a given animal	**Collaboration:** Animal

Classname: ControlPanel	
Superclass: JPanel **Subclass(es):** none	
Responsibilities: The class groups buttons needed to control the simulation.	**Collaboration:** SimulationFrame

Classname: FieldPanel	
Superclass: JPanel **Subclass(es):** none	
Responsibilities: Single field panel. Groups the field information and displays the field.	**Collaboration:** AnimalStatsFrame, Field

Classname: LegendPanel	
Superclass: JPanel **Subclass(es):** none	
Responsibilities: Helps the user understand the meaning of individual fields and colors.	**Collaboration:** none

Classname: ParametersFrame	
Superclass: JFrame **Subclass(es):** none	
Responsibilities: Class that displays a window in which the user can set initial parameters of the simulation.	**Collaboration:** Parameters, StartPanel

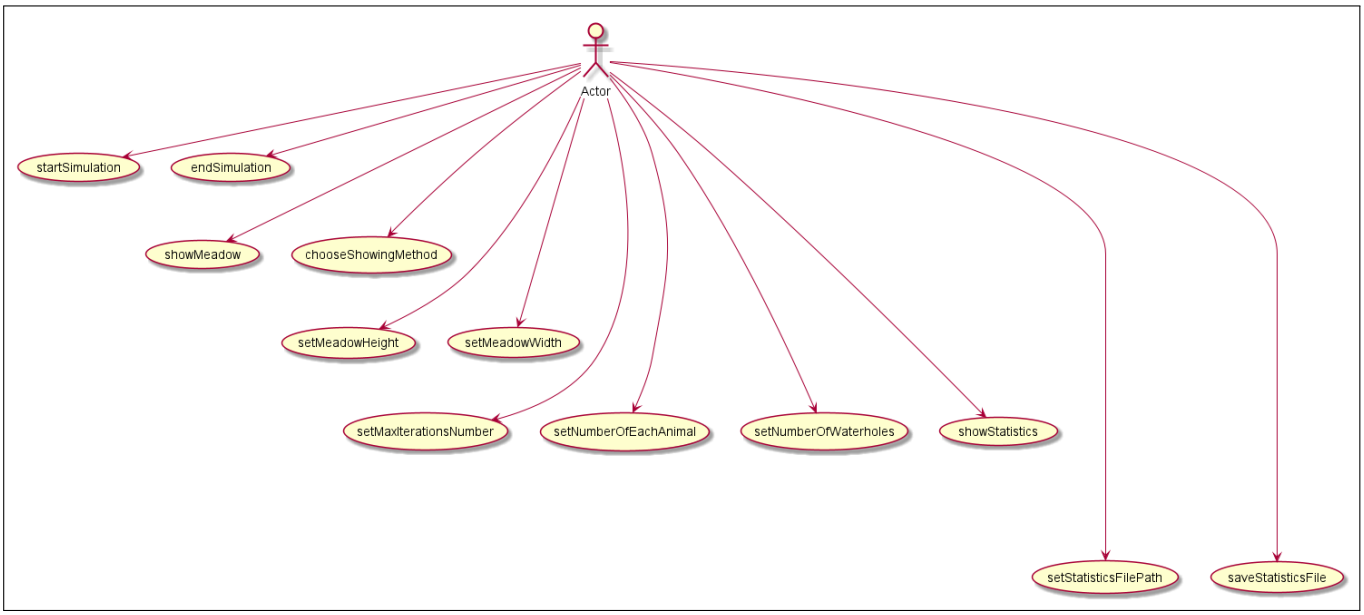
Classname: SimulationFrame	
Superclass: JFrame **Subclass(es):** none	
Responsibilities: Main simulation window. Groups the panels necessary to run simulation: legend panel, panels of individual fields, control panel and statistics panel.	**Collaboration:** Simulation, Parameters, StartFrame, StatsPanel, LegendPanel, FieldPanel, ControlPanel

Classname: StartFrame	
Superclass: JFrame **Subclass(es):** none	
Responsibilities: Starting simulation window. Displays the start panel. It allows user to display program information, display parameter window and run simulation.	**Collaboration:** Parameters, StartPanel, ParametersFrame, Main, SimulationFrame

Classname: StartPanel	
Superclass: JPanel **Subclass(es):** none	
Responsibilities: The start panel groups 3 buttons: start simulation, configuration and project information. Clicking on one of them calls the appropriate actions.	**Collaboration:** StartFrame, ParametersFrame

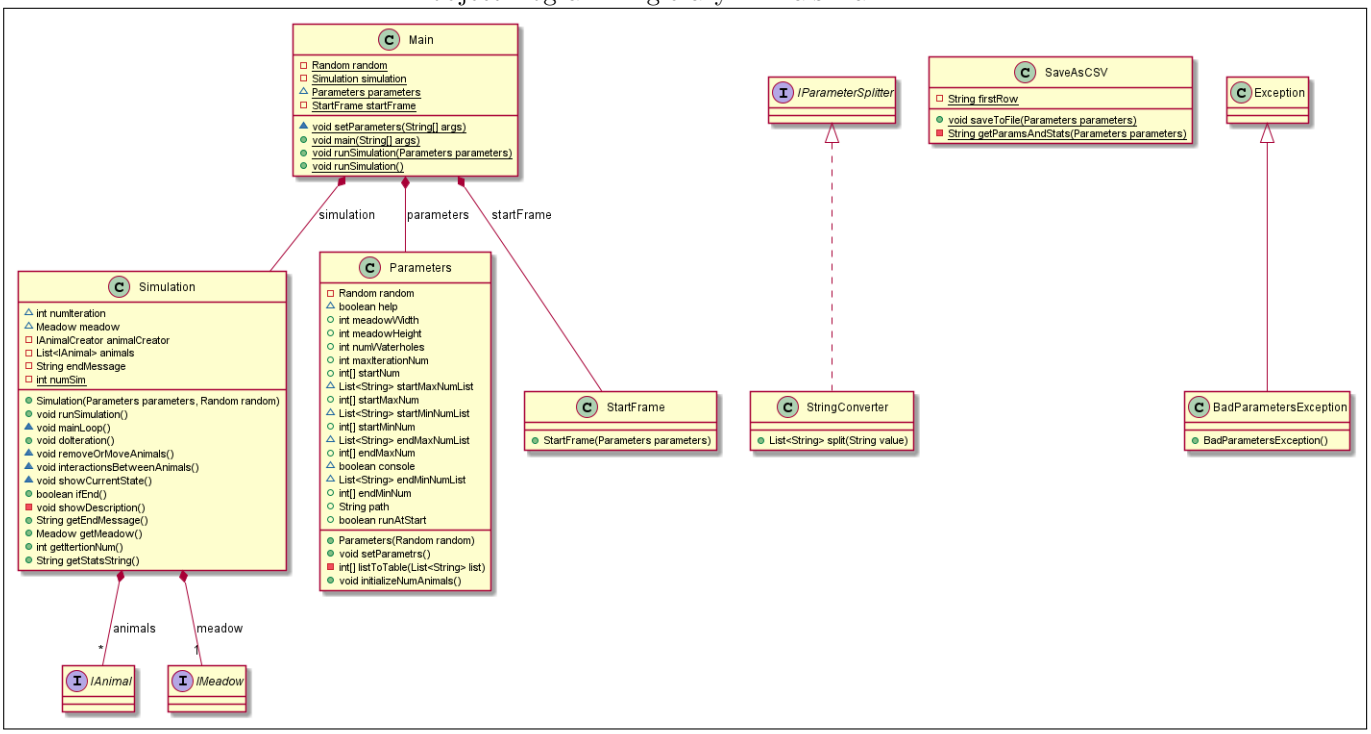
Classname: StatsPanel	
Superclass: JPanel **Subclass(es):** none	
Responsibilities: A panel displaying statistics: iteration number, number of animals and food for each species.	**Collaboration:** SimulationFrame

3 Diagram przypadków użycia

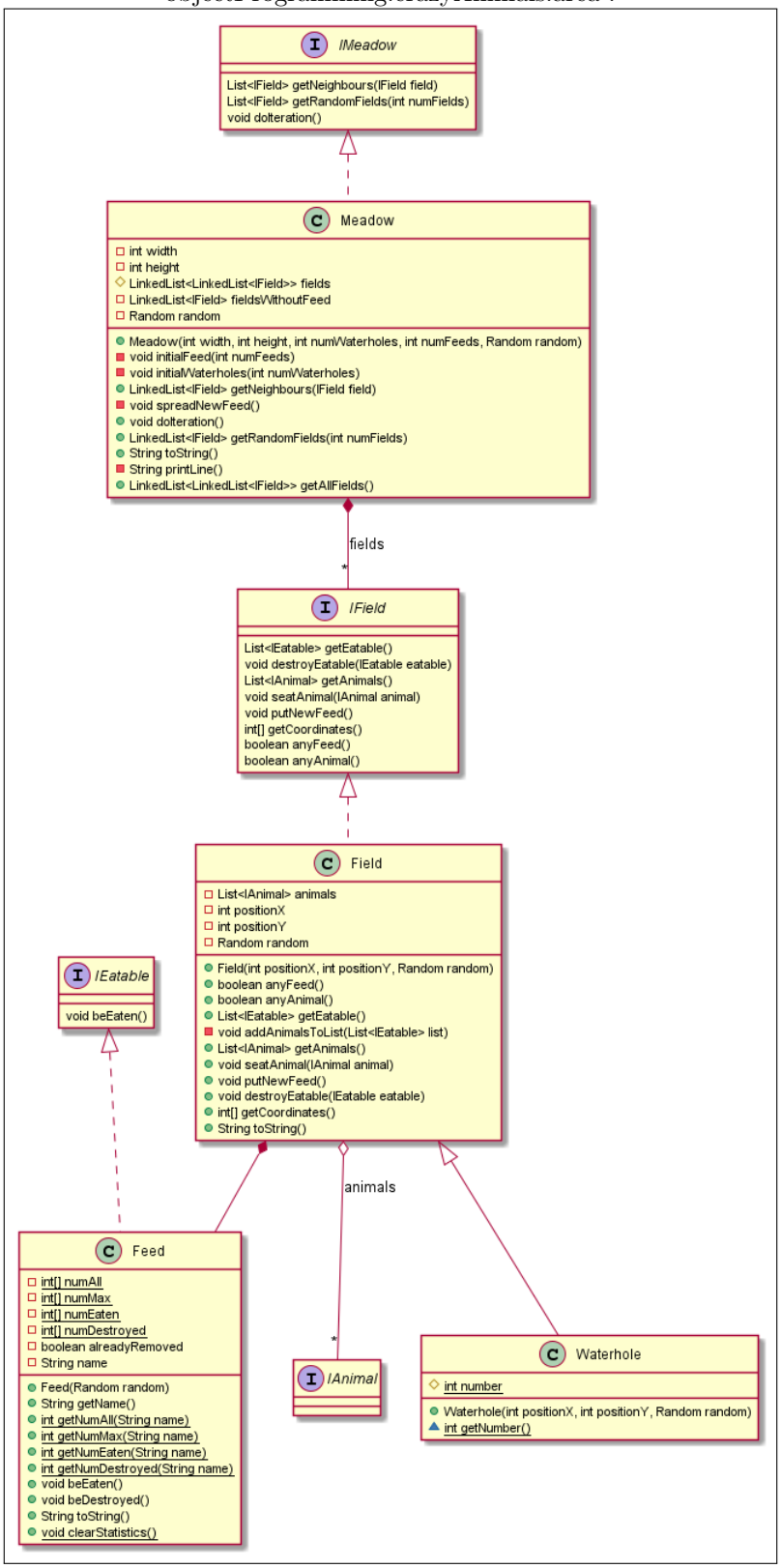


4 Diagramy klas

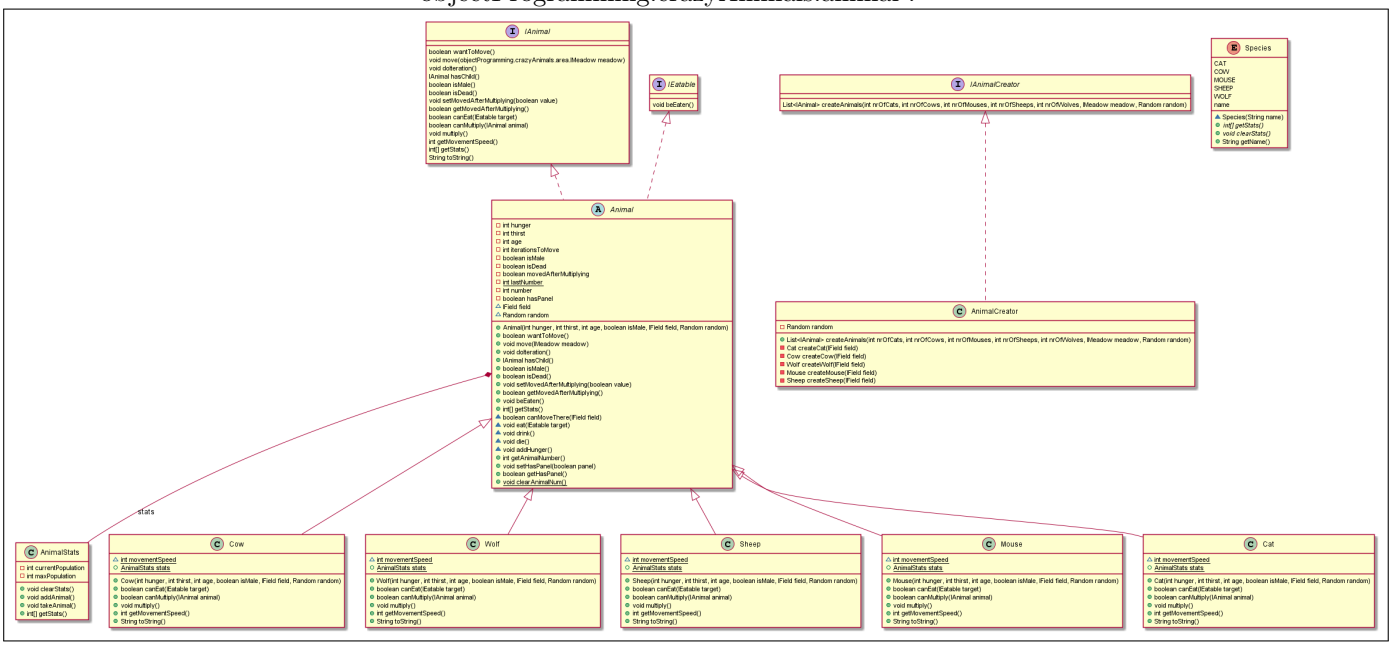
objectProgramming.crazyAnimals.main :



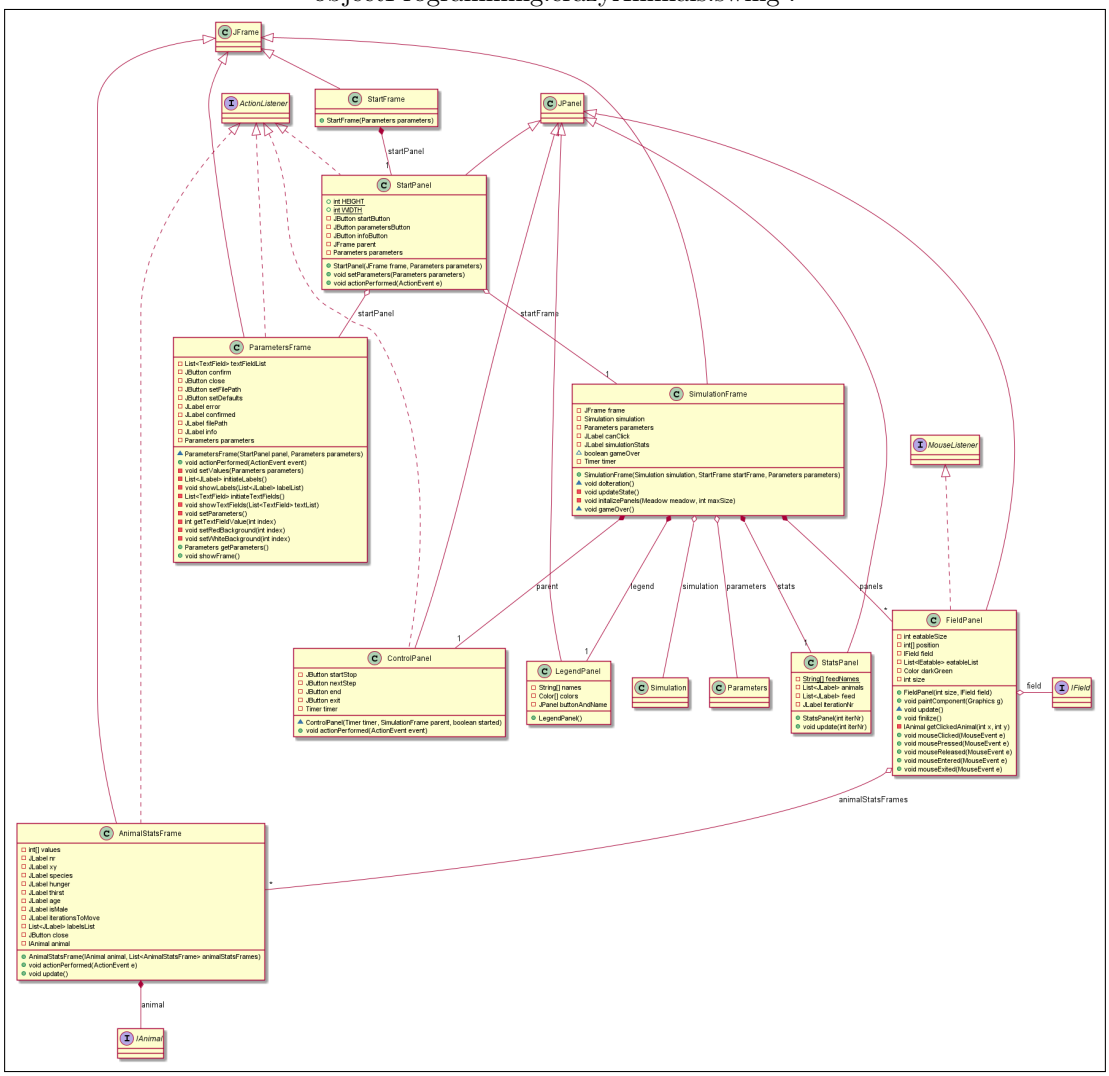
objectProgramming.crazyAnimals.area :



objectProgramming.crazyAnimals.animal :

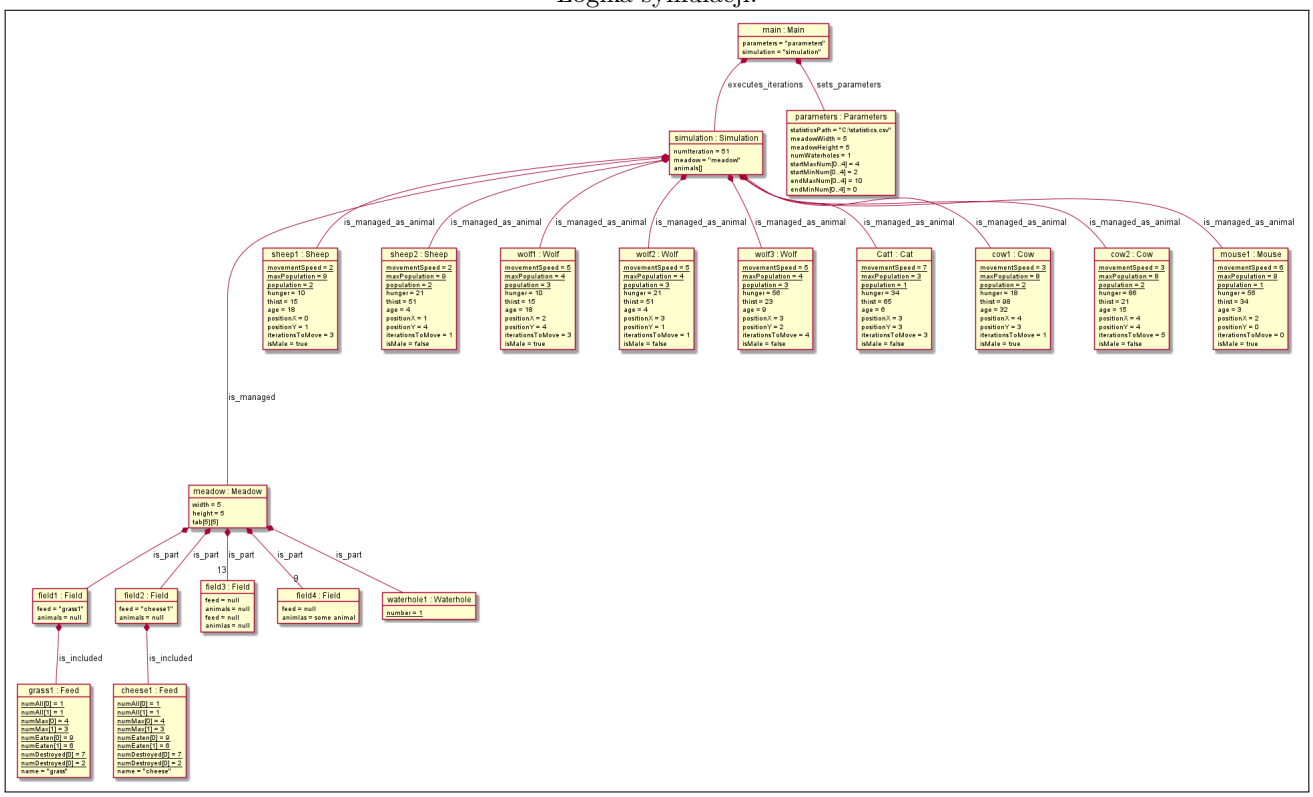


objectProgramming.crazyAnimals.swing :

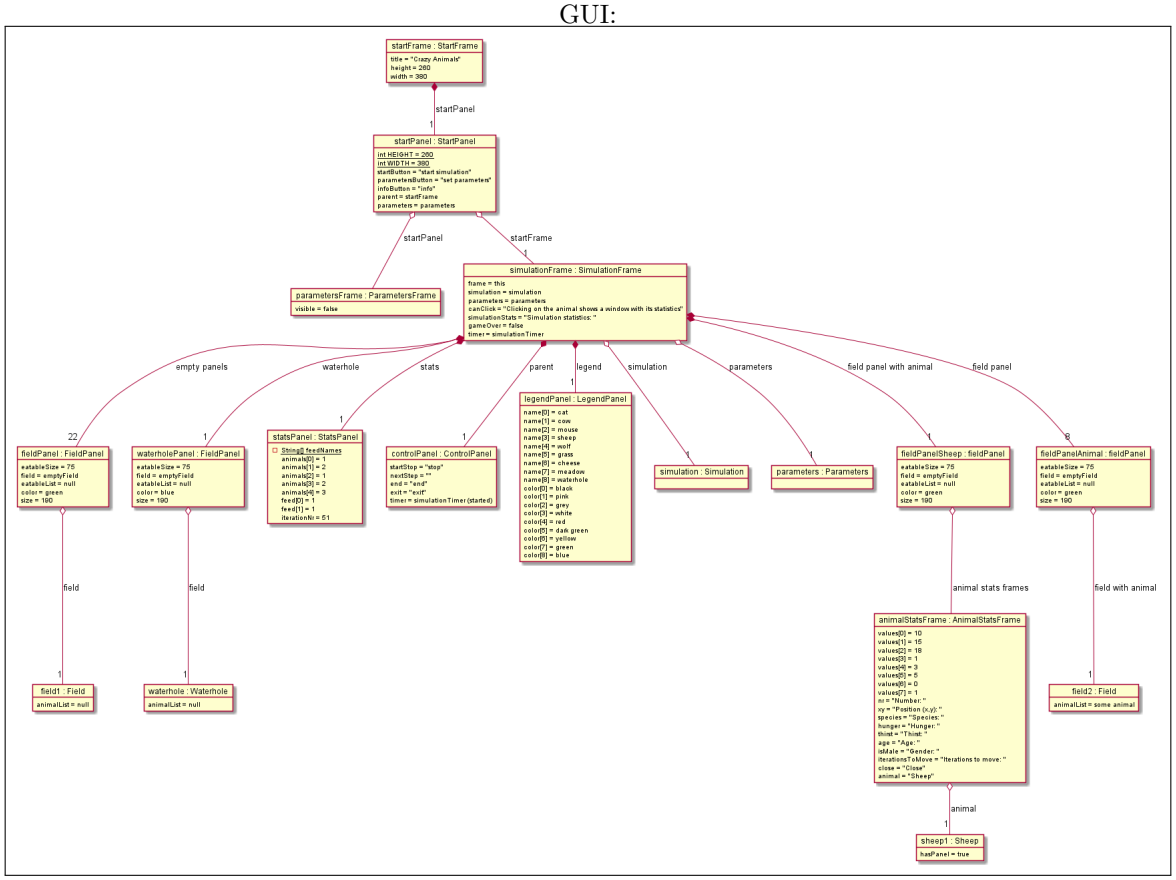


5 Diagramy obiektów

Logika symulacji:

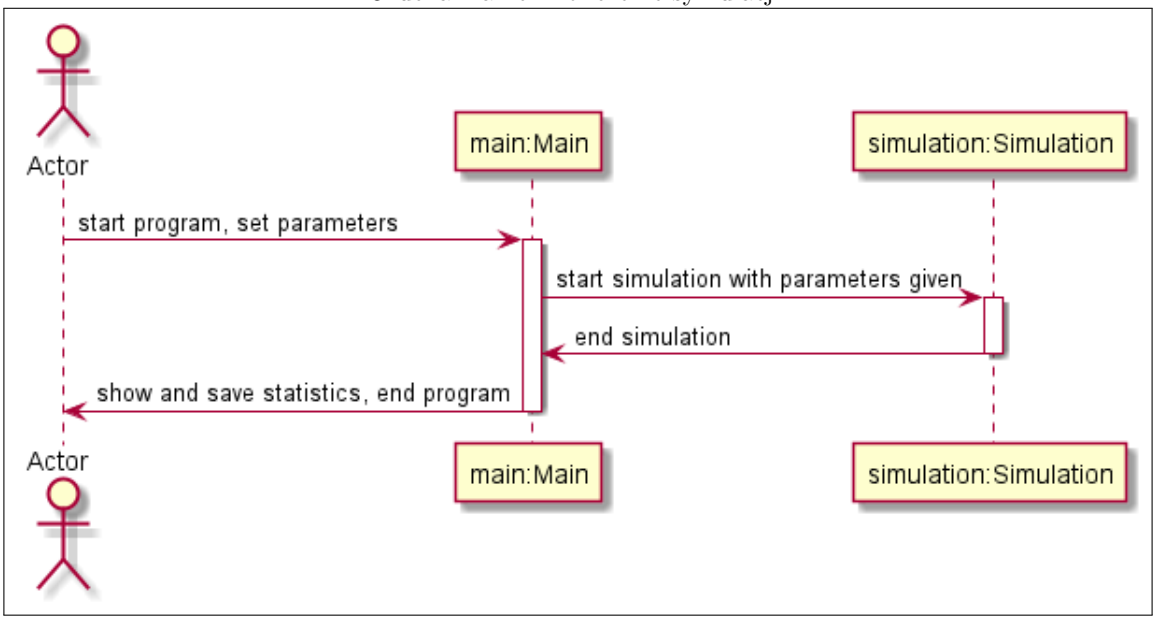


GUI:

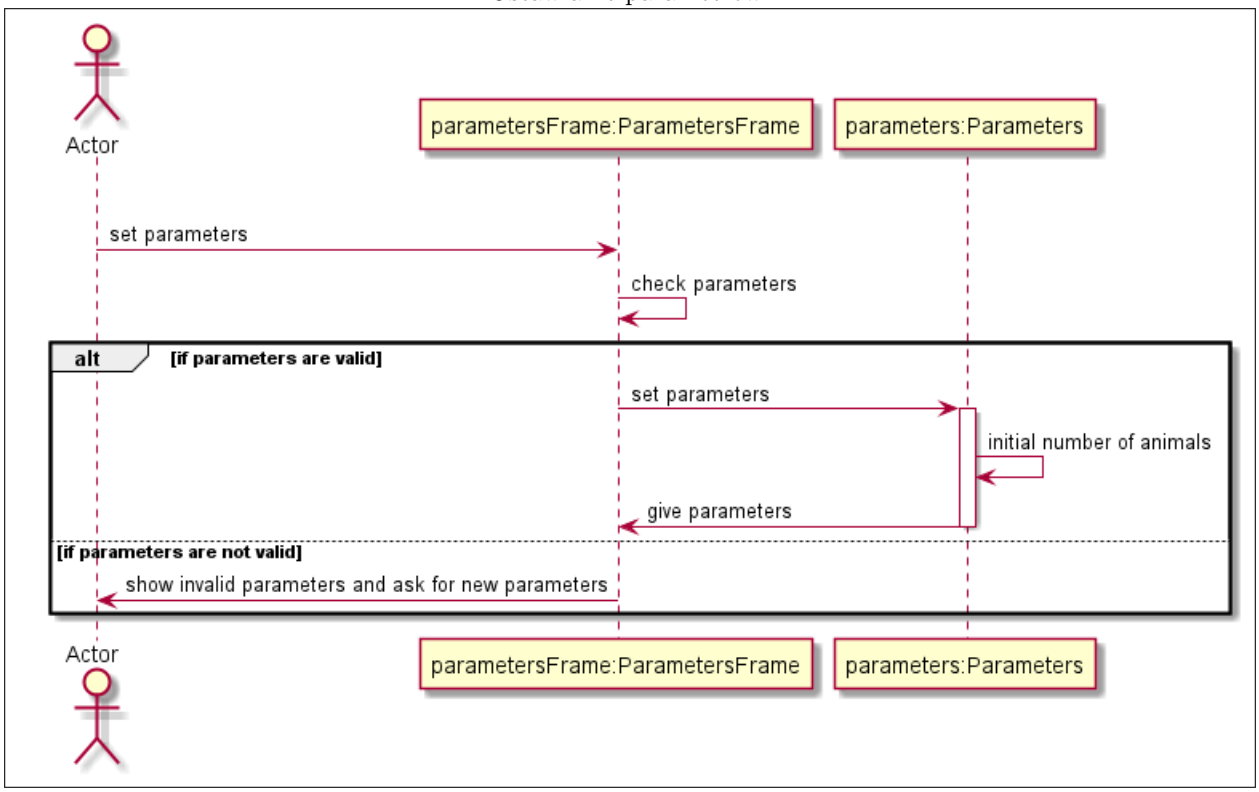


6 Diagramy sekwencji

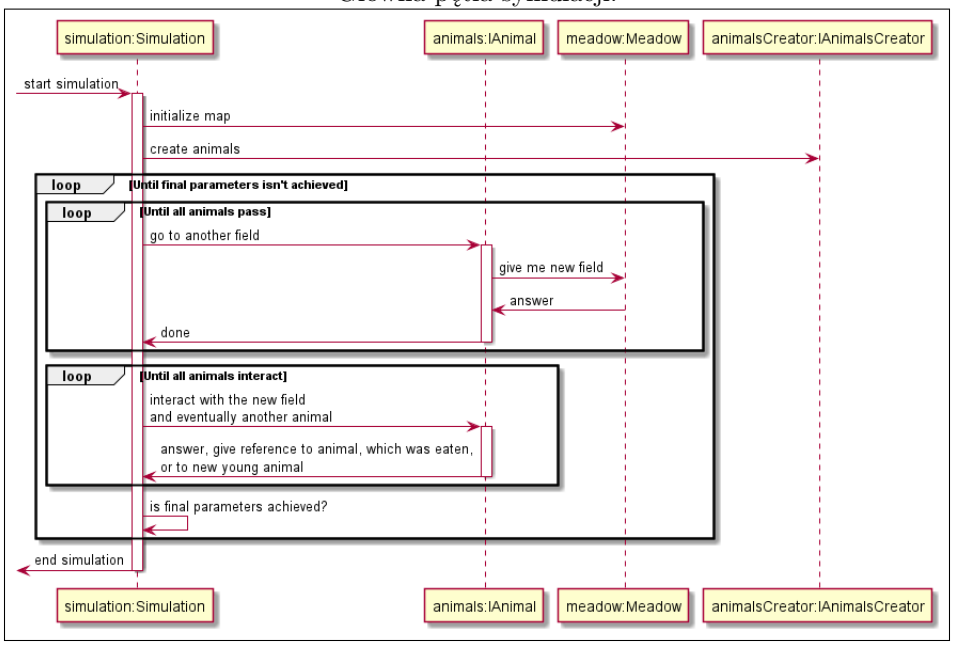
Uruchamianie i kończenie symulacji:



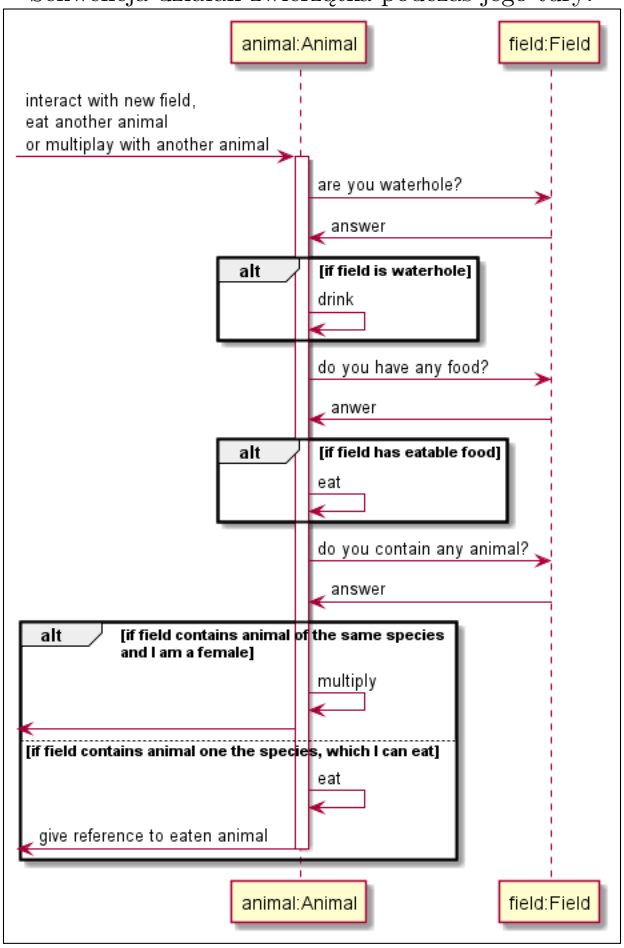
Ustawianie parametrów:



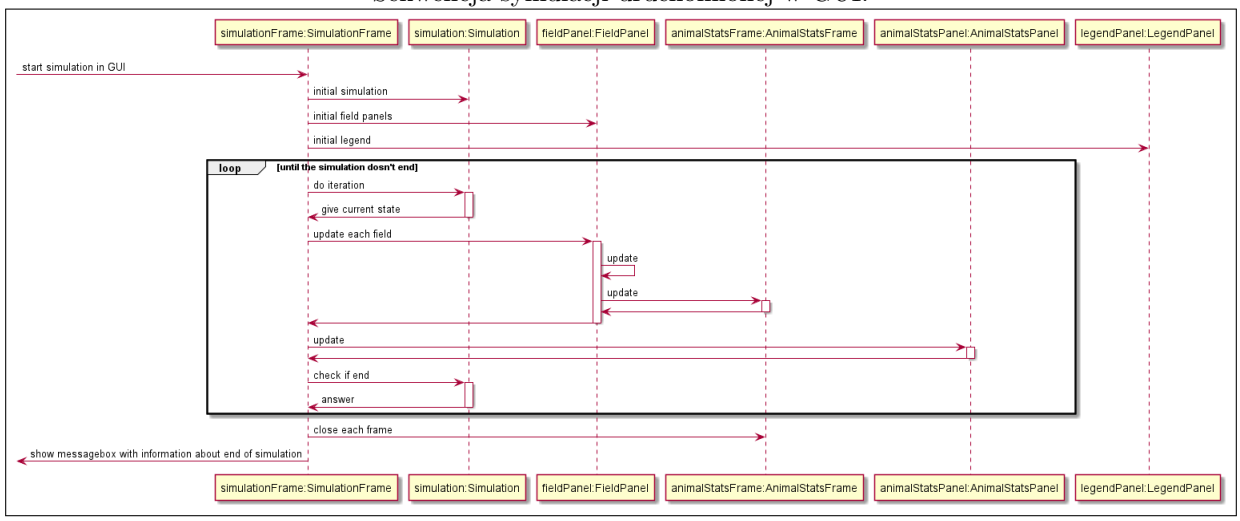
Główna pętla symulacji:



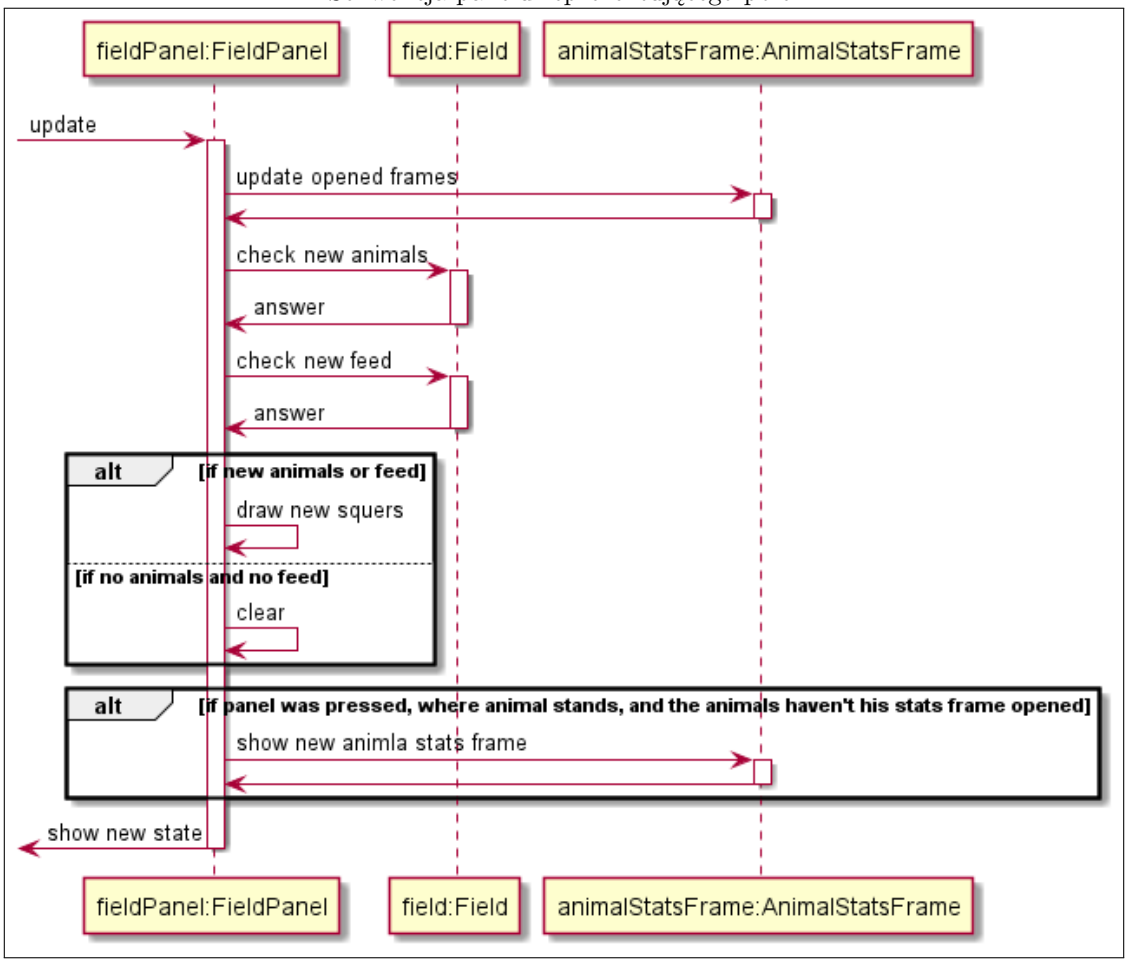
Sekwencja działań zwierzątka podczas jego tury:



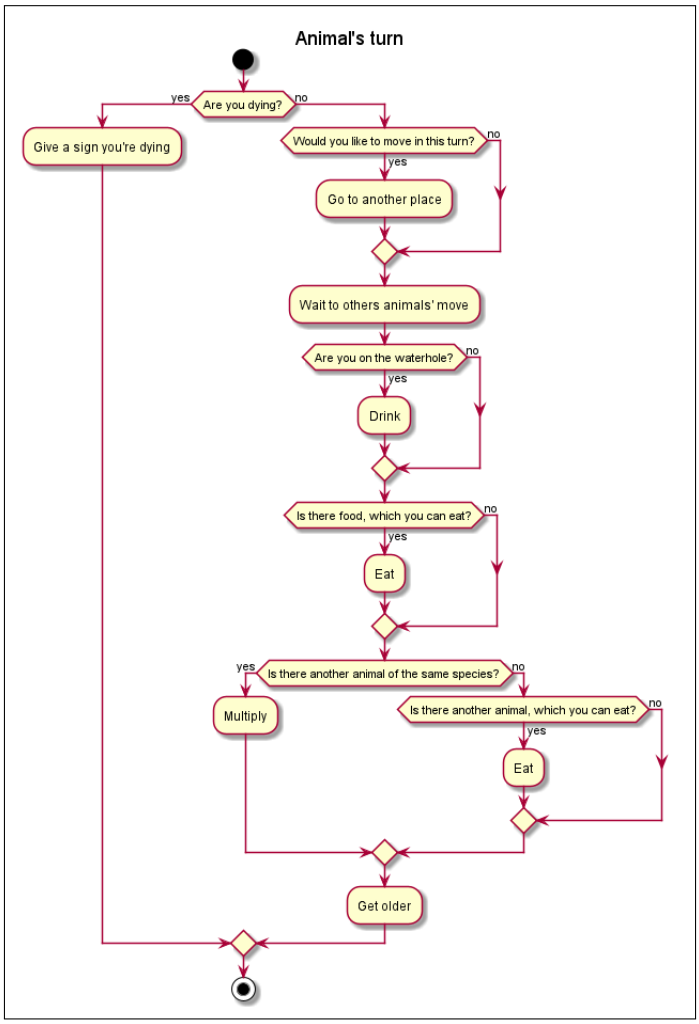
Sekwencja symulacji uruchomionej w GUI:

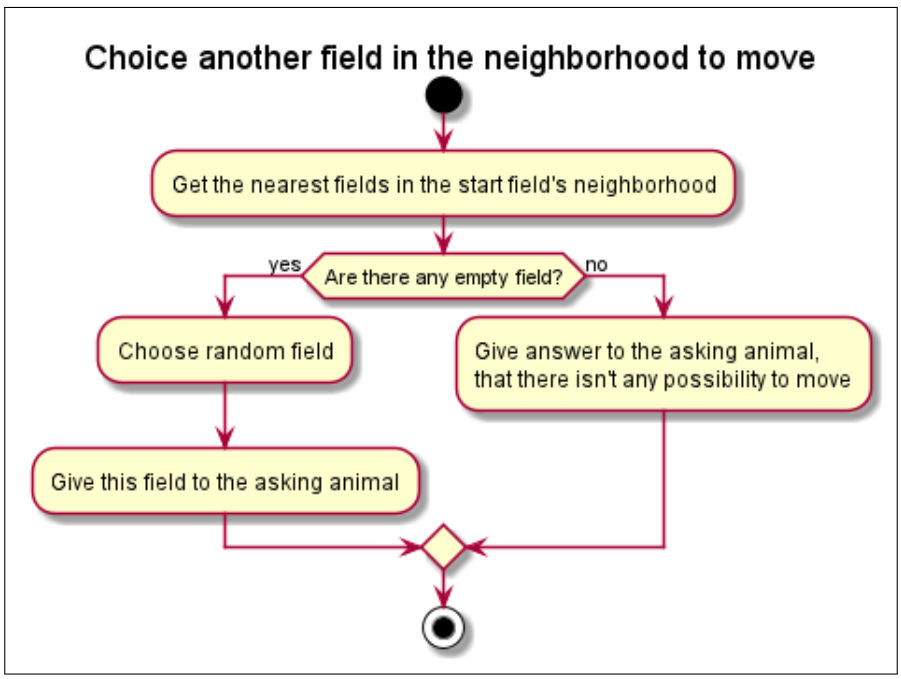


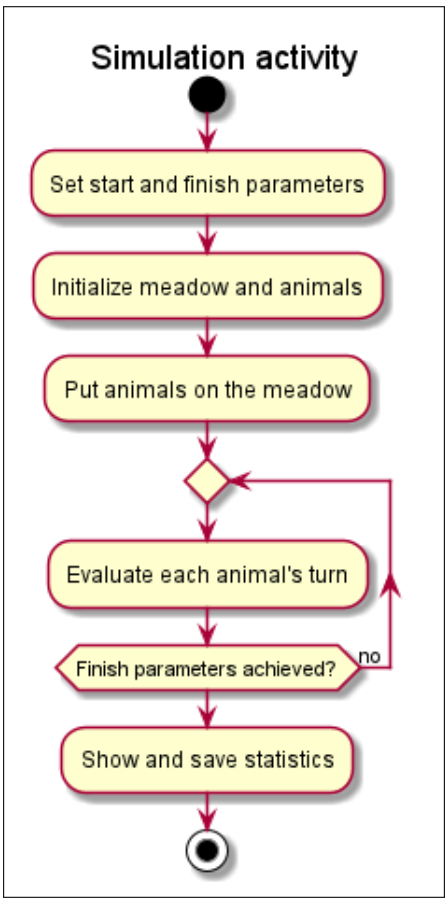
Sekwencja panelu reprezentującego pole:

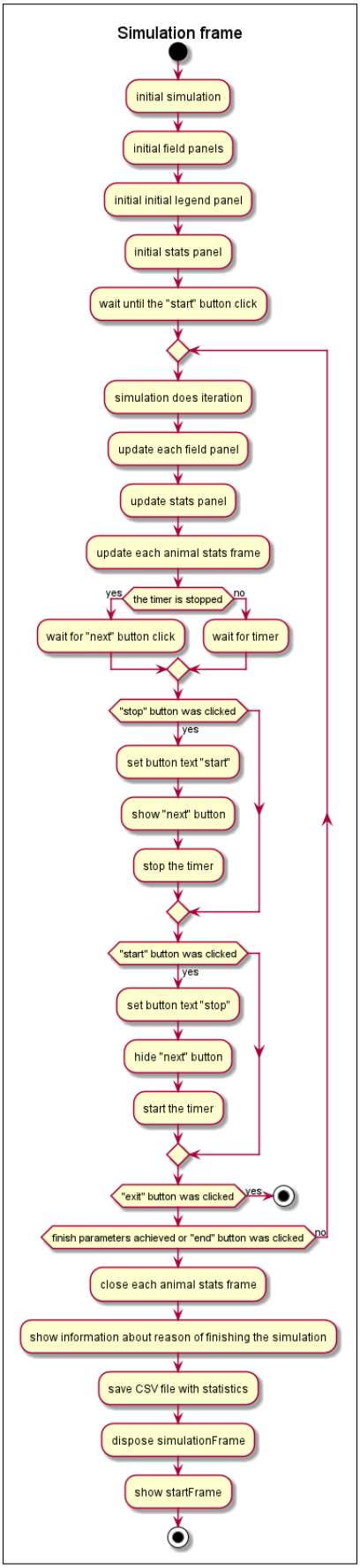


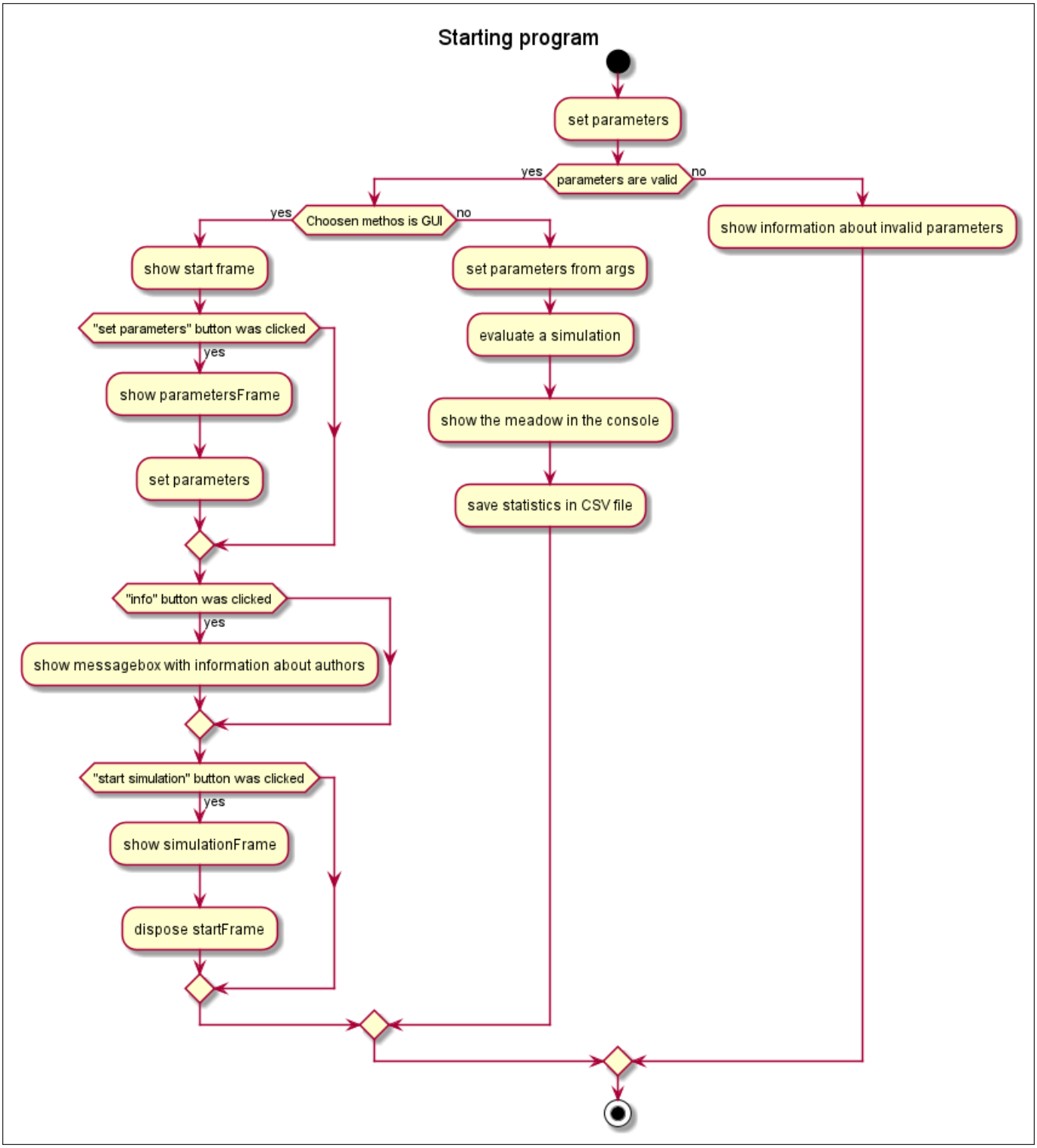
7 Diagramy aktywności



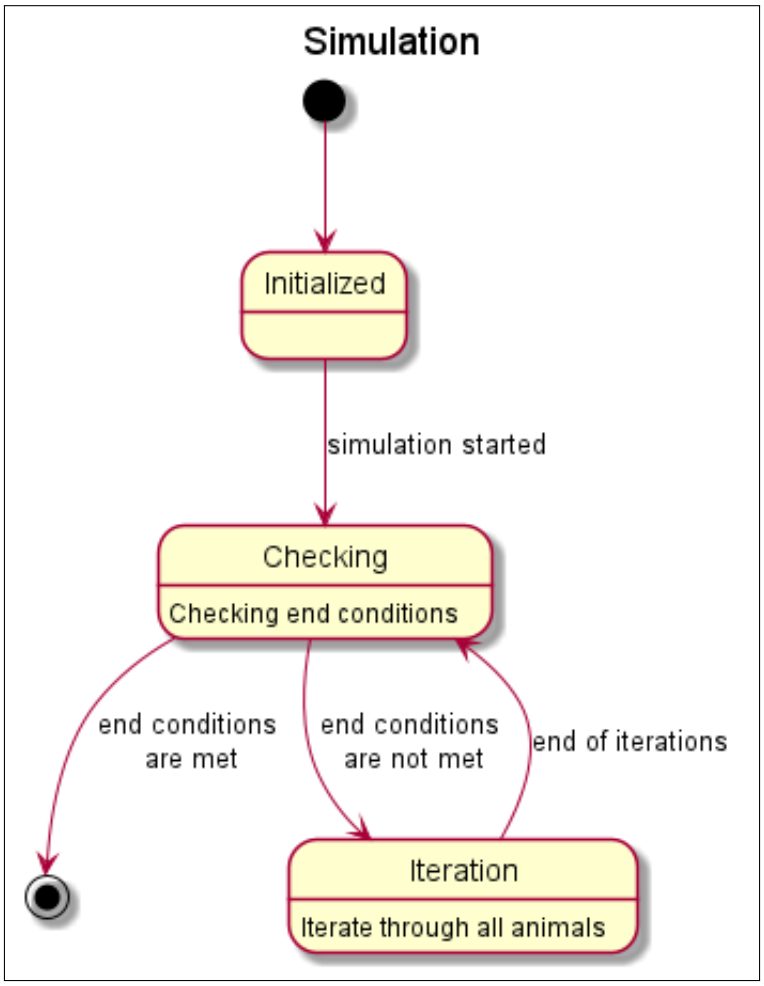


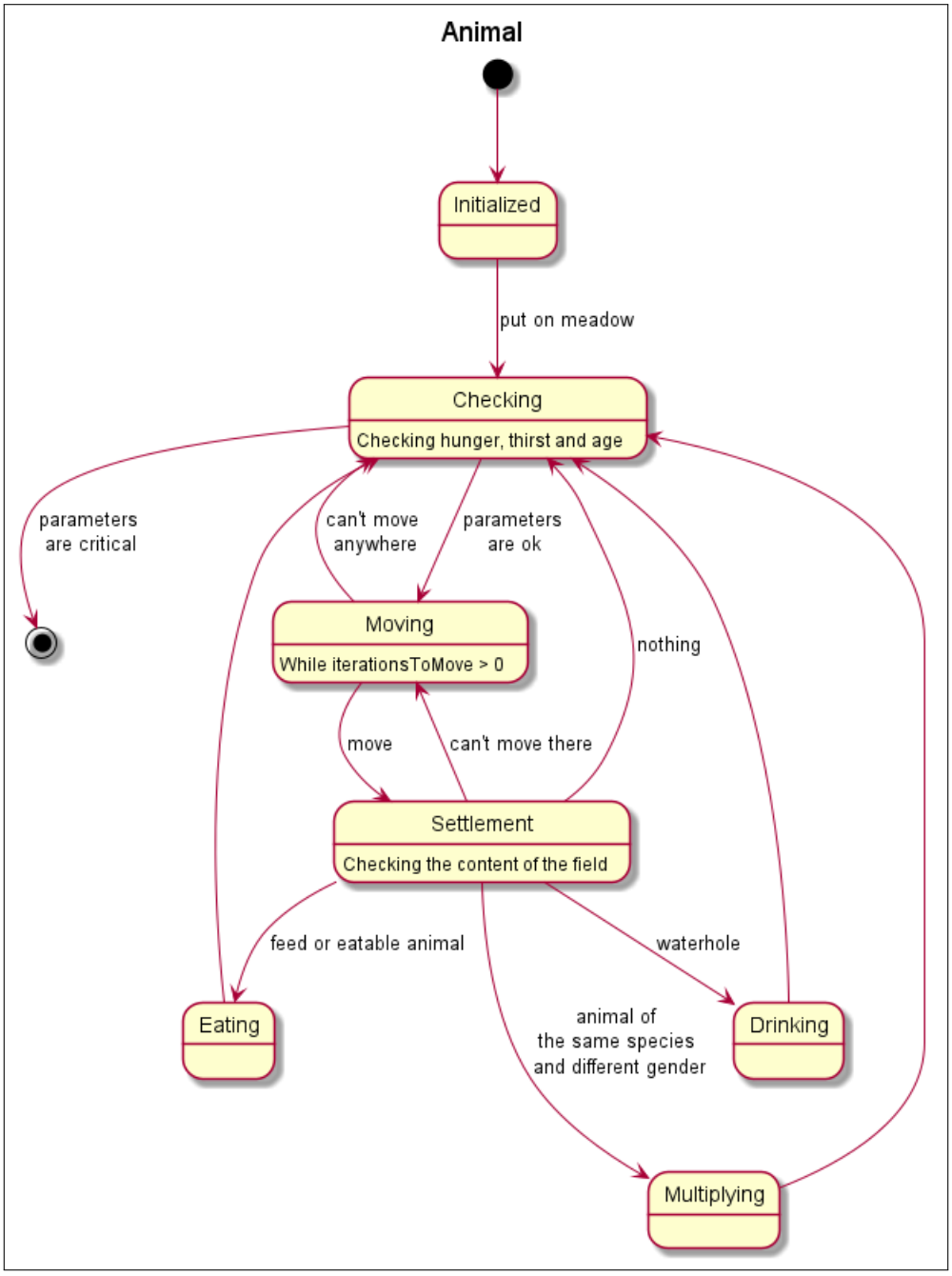

Simulation activity
Set start and finish parameters
Initialize meadow and animals
Put animals on the meadow
Evaluate each animal's turn
Finish parameters achieved?
no
Show and save statistics

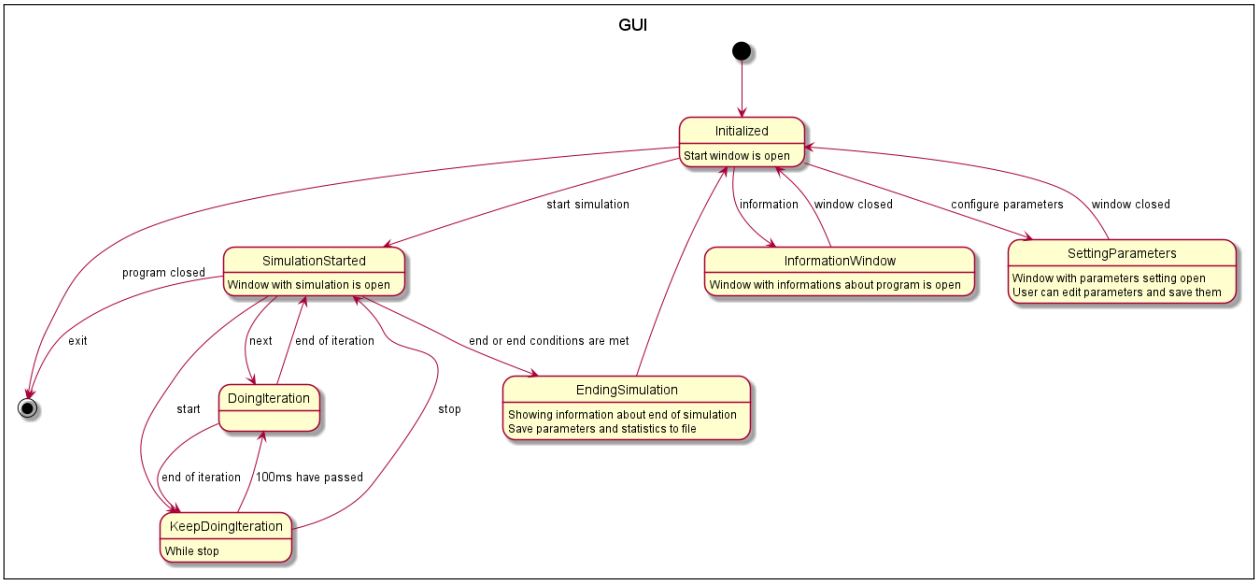

Simulation frame
initial simulation
initial field panels
initial initial legend panel
initial stats panel
wait until the "start" button click
simulation does iteration
update each field panel
update stats panel
update each animal stats frame
the timer is stopped
yes
no
wait for "next" button click
wait for timer
"stop" button was clicked
yes
set button text "start"
show "next" button
stop the timer
"start" button was clicked
yes
set button text "stop"
hide "next" button
start the timer
"exit" button was clicked
yes
finish parameters achieved or "end" button was clicked
no
close each animal stats frame
show information about reason of finishing the simulation
save CSV file with statistics
dispose simulationFrame
show startFrame


Starting program
set parameters
parameters are valid
yes
no
show information about invalid parameters
Choosen methos is GUI
yes
no
show start frame
set parameters from args
"set parameters" button was clicked
yes
evaluate a simulation
show parametersFrame
show the meadow in the console
set parameters
save statistics in CSV file
"info" button was clicked
yes
show messagebox with information about authors
"start simulation" button was clicked
yes
show simulationFrame
dispose startFrame

8 Diagramy maszyny stanów




Animal
Initialized
put on meadow
Checking
Checking hunger, thirst and age
parameters
are critical
can't move
anywhere
parameters
are ok
Moving
While iterationsToMove > 0
nothing
move
can't move there
Settlement
Checking the content of the field
feed or eatable animal
waterhole
Eating
animal of
the same species
and different gender
Drinking
Multiplying



9 Link do repozytorium

`https://github.com/mikolajchmielecki/CrazyAnimals.git`