Programowanie Obiektowe - Projekt
Etap 3

Mikołaj Chmielecki, Jakub Mroziński

30 kwietnia 2020

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 we wszystkich przypadkach: z myszą – 100%, z kotem – 80%, z owcą – 60%, z krową – 40%. Jeżeli atak skończy się niepowodzeniem to zwierzęta rozchodzą się osłabione. 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 wszystkie zwierzęta zginą lub gdy któryś z gatunków osiągnie ustaloną przed rozpoczęciem symulacji liczebność. 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 and collects statistics for each species (current and maximum population).	**Collaboration:** Simulation

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:** none

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:** none

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:** none

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:** none

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:** none

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: 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.	**Collaboration:** Feed, Meadow

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

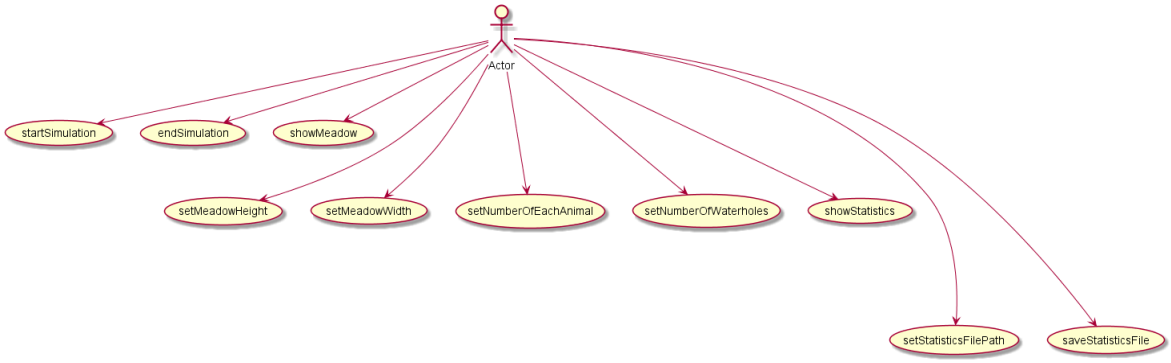
Classname: Feed	
Superclass: none **Subclass(es):** none	
Responsibilities: The class stores information about food placed on the meadow. It also contains statistics on the amount of food eaten and destroyed during stimulation.	**Collaboration:** Field

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

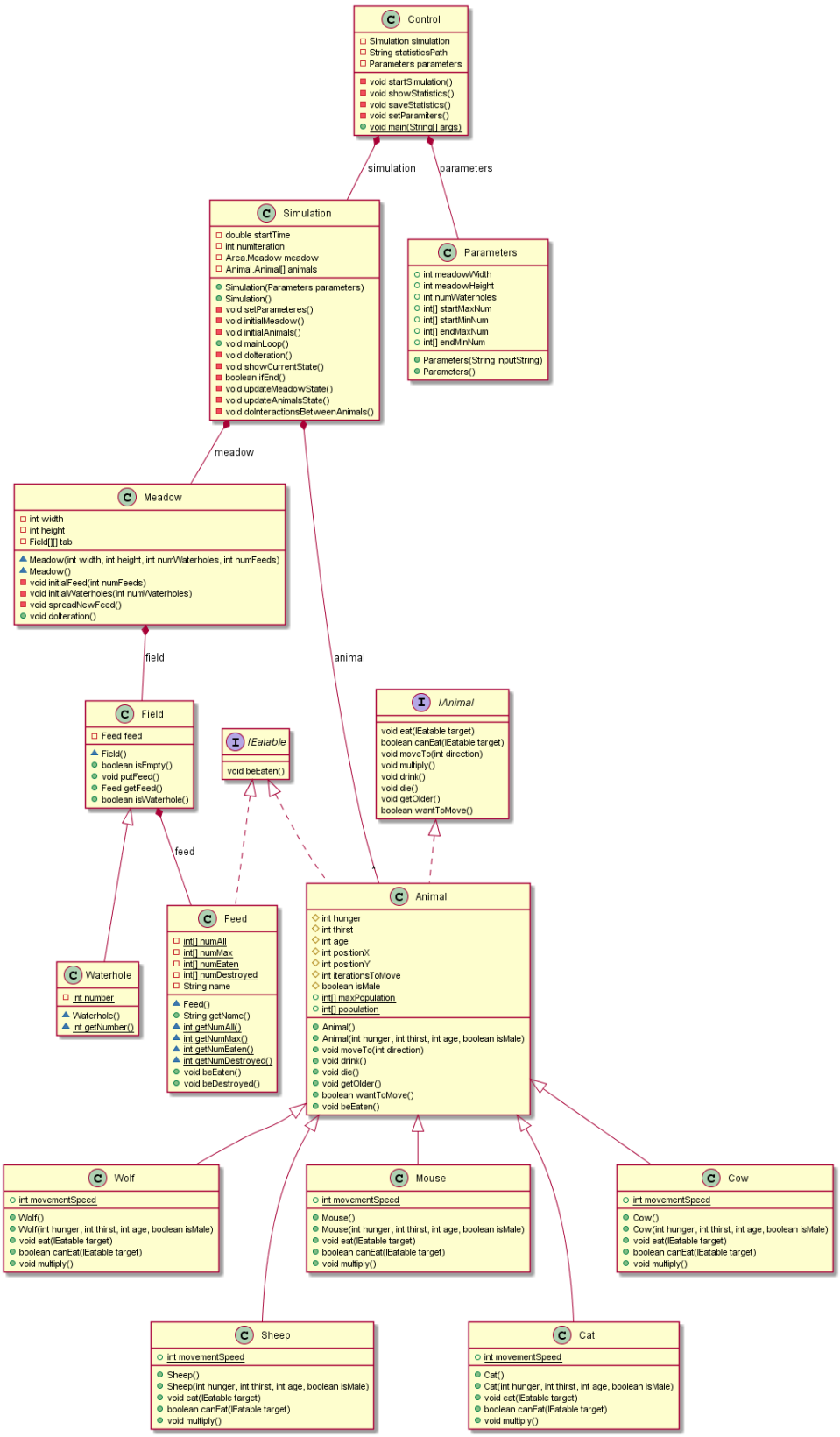
Classname: Simulation	
Superclass: none **Subclass(es):** none	
Responsibilities: The class is responsible for the simulation. In it there is the main simulation loop. It coordinates the actions of animals and forces interactions between them, such as reproduction, quenching thirst or hunger. Generates 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, Control

Classname: Control	
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

3 Diagram przypadków użycia



4 Diagram klas



5 Diagram obiektów