

Programowanie Obiektowe

Kierunek

Informatyka Techniczna

Termin

Środa TN 15:15

Imię, nazwisko, numer albumu

Bartosz Ostrowski 259222

Data

24 czerwca 2021

Link do projektu

<https://www.overleaf.com/read/zvpwwmddrpk>



RAPORT

1 Jurassic Park



1.1 Ogólne przedstawienie

Symulacja będzie swoistą próbą przedstawienia filmowego Parku Jurajskiego, z głównym nastawieniem na cykl życia dinozaurów - podział na roślinożerność, wszystkożerność i mięsożerność. Ograniczoną listę dinozaurów opartą na encyklopedii (ów encyklopedia będzie zawarta w kodzie symulacji, będzie to lista określonych gatunków importowana z pliku dinosaurs.json). Plansza na której będzie się rozgrywać symulacja będzie losowo generować określone dinozaury. Na planszy dla roślinożerców powstanie obiekt będący reprezentacją ich jedzenia. Powstanie również obiekt wody - do napełniania pragnienia.

1.2 Wartości startowe

Będzie można określić prawdopodobieństwo wylosowania dinozaura i obiektu. Maksymalny rozmiar parku. Będzie też możliwa do ustawienia ilość iteracji, ile razy ma zostać wykonana cała plansza.

1.3 Wstępne klasy

1. **Dinozaur** - będący główną klasą, mogący zawierać takie wartości jak:

- gatunek z encyklopedii
- sposób odżywiania się (roślinożerność, wszystkożerność i mięsożerność)
- zdrowie - gatunki mięsożerne będą musiały polować, przykładowo zdrowie 0 oznacza zwłoki/padlinę występującą na planszy
- głód - w przypadku spadnięcia do 0, dostaje obrażenia w każdej następnej iteracji

- pragnienie - uzupełniane przy złożach wody na planszy, w przypadku spadnięcia do 0 dostaje obrazienia w każdej następnej iteracji
 - szybkość - każdy dinozaur charakteryzuje się inną szybkością
2. **Rośliny** - obiekt głównie do jedzenia dla roślinożerców:
- ilość w jednym obiekcie
3. **Woda** - obiekt głównie do napełniania pragnienia dla wszystkich typów dinozaurów
- ilość w jednym obiekcie

1.4 Koniec symulacji

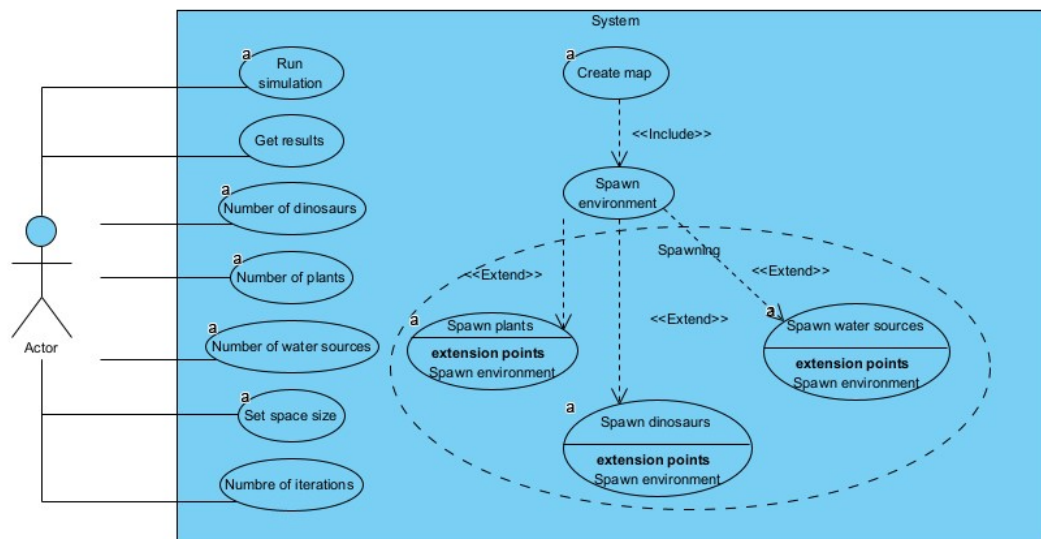
Symulacja kończy się po określonej liczbie iteracji.

2 Analiza czasownikowo-rzeczownikowa

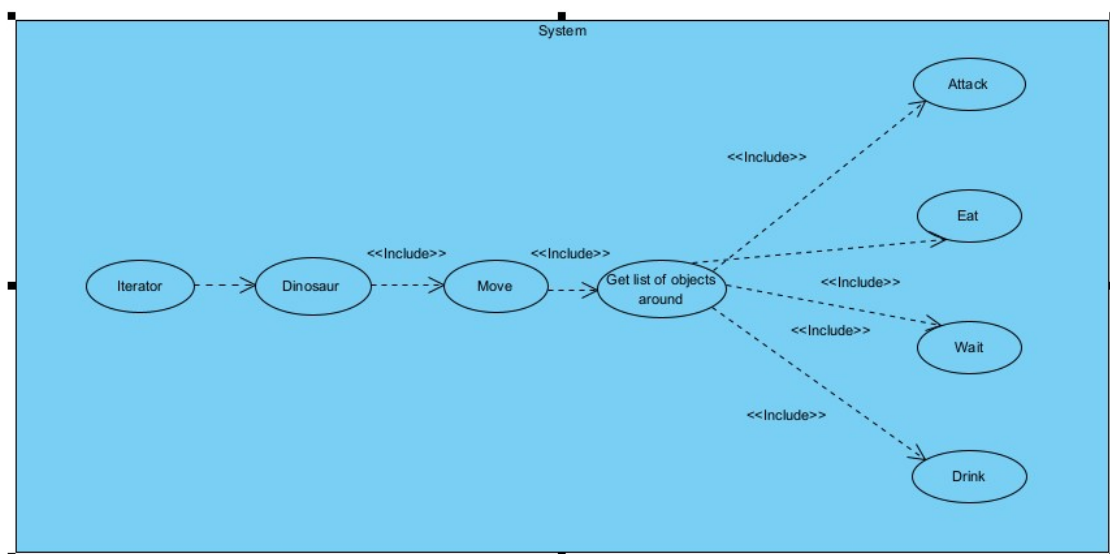
Projektujemy symulację agentową, która będzie badać zachowanie dinozaurów różnych gatunków. Symulacja będzie zawierać planszę dwuwymiarową (X,Y) o zadanej wielkości, gdzie na każdym jednym polu powstanie obiekt rośliny, wody, dinozaura, bądź pustego pola (tzw. Tile).

- Zachowanie się dinozaurów:
 - W zależności od gatunku dinozaur może jeść określone pożywienie - rośliny, albo innego dinozaura (gdzie dinozaur "je" innego, tylko podczas ataku, zwracana jest adekwatna ilość głodu do ataku, dinozaur może atakować martwego, co też daje mu pożywienie), bądź to i to, zaspelniając w ten sposób głód.
 - Mięsożercy, oraz wszystkożercy mogą atakować dinozaury innego gatunku, co skutkuje obniżeniem zdrowia atakowanego dinozaura o określoną wartość i uzupełnia lekko głód atakującego.
 - Podobnie z pragnieniem, tylko uzupełnianie go będzie przy polach wód.
 - Na starcie symulacji dinozaury pojawiają się w losowych polach.
 - Dinozaury poruszają się w losowym kierunku, o określoną w gatunku liczbę pól. Mogą dostać się tylko na nieoblegane pole i wykonać możliwą akcję z polem sąsiadującym PO RUCHU.
- Pola rośliny, wód i pól (tzw. Tile) są statyczne, nie poruszają się, mogą być tylko użyte przez dinozaury.
- Parametry symulacji:
 - Wielkość przestrzeni S*S.
 - Importowana z pliku dinosaurs.json - n liczba gatunków
 - Prawdopodobieństwo pojawienia się elementu (element, to statyczne obiekty czyli roślina, woda bądź puste pole) E_r .
 - Prawdopodobieństwo pojawienia się dinozaura (losowany jest z importowanej listy (encyklopedii), więc nie ma zasady pojawienia się więcej danego typu) D_r .
 - Gdzie $\sum E_r + D_r + x = S * S$, gdzie x to pozostałe puste niezamieszkane pola.
 - Maksymalna liczba iteracji I.

3 Diagramy UC



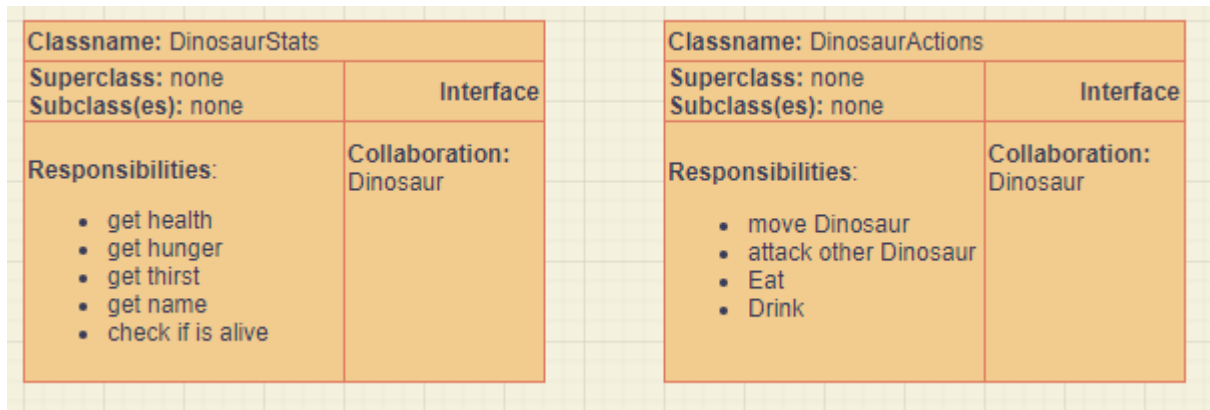
Rysunek 1: Diagram UC - ustawienia i opcje użytkownika oraz tworzenie mapy.



Rysunek 2: Diagram UC - poglądowe działanie systemu.

4 Karty CRC

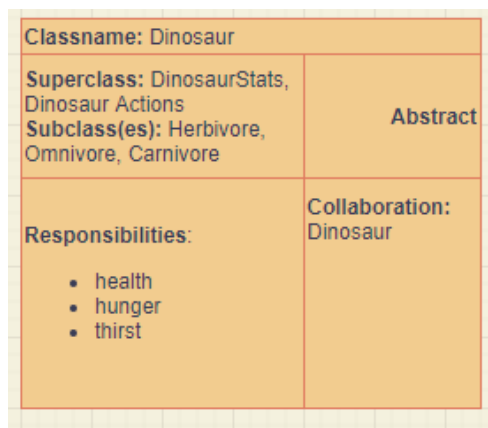
4.1 Interfejsy



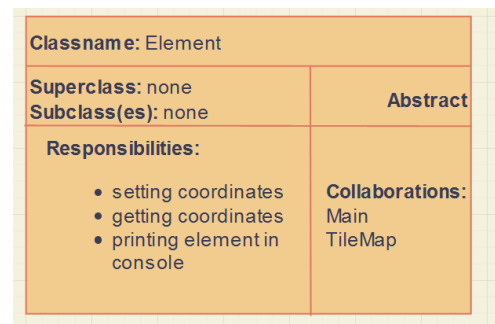
Rysunek 3: Interfejsy do obsługi metod klasy Dinosaur.

4.2 Klasy abstrakcyjne

W tym wypadku klasa abstrakcyjna Dinosaur może dziedziczyć po dwóch interfejsach (zgodnie z dokumentacją Javy - Klasa może dziedziczyć tylko po jednej Klasie, ale interfejsy są wyjątkiem).



Rysunek 4: Klasa abstrakcyjna Dinosaur.



Rysunek 5: Klasa abstrakcyjna Element.

4.3 Klasy

Classname: Herbivore		Classname: Omnivore		Classname: Carnivore	
Superclass: Dinosaur	Class	Superclass: Dinosaur	Class	Superclass: Dinosaur	Class
Subclass(es): none		Subclass(es): none		Subclass(es): none	
Responsibilities: <ul style="list-style-type: none">• speed• checks which food can be eaten• get objects around	Collaboration: WaterSource, Plant, Dinosaur	Responsibilities: <ul style="list-style-type: none">• speed• checks which food can be eaten• get objects around	Collaboration: WaterSource, Plant, Dinosaur	Responsibilities: <ul style="list-style-type: none">• speed• checks which food can be eaten• get objects around	Collaboration: Map, WaterSource, Plant, Dinosaur

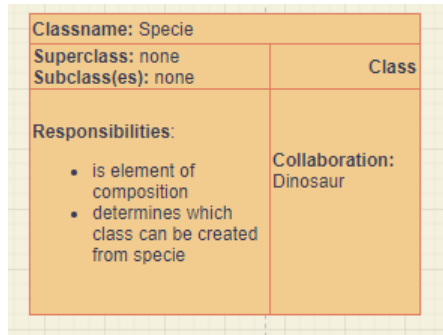
Rysunek 6: Klasy dziedziczące po klasie abstrakcyjnej Dinosaur.

Classname: WaterSource		Classname: Plant	
Superclass: Element	Class	Superclass: Element	Class
Subclass(es): none		Subclass(es): none	
Responsibilities: <ul style="list-style-type: none">• object on map• interacts with Dinos• capacity• checks if is empty• interacts with other objects of WaterSource class	Collaboration: Dinosaur	Responsibilities: <ul style="list-style-type: none">• object on map• interacts with Dinos• name• gets value of plant• gets toxicity of plant	Collaboration: Dinosaur

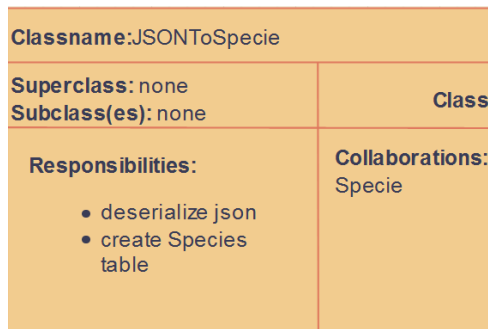
Rysunek 7: Klasy będące elementami mapy.

Classname: Tile	
Superclass: Element	Class
Subclass(es): none	
Responsibilities: <ul style="list-style-type: none">• represents empty tile to move on• static	Collaborations: Main TileMap

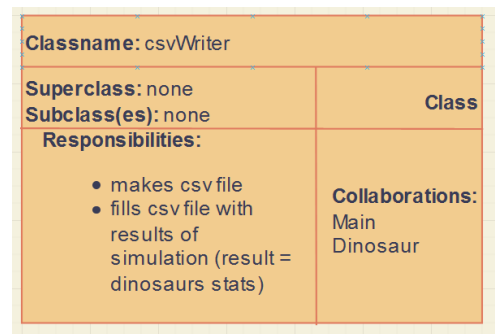
Rysunek 8: Klasa będąca elementem mapy.



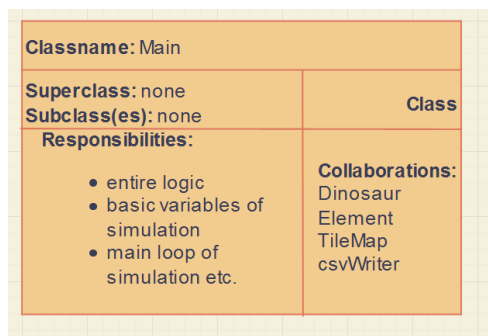
Rysunek 9: Klasa która zawiera się w klasie abstrakcyjnej Dinosaur (kompozycja).



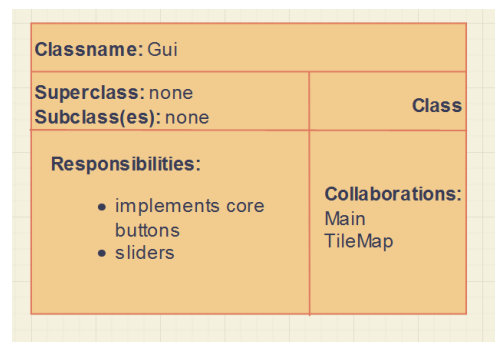
Rysunek 10: Klasa do obsługi deserializacji plików JSON.



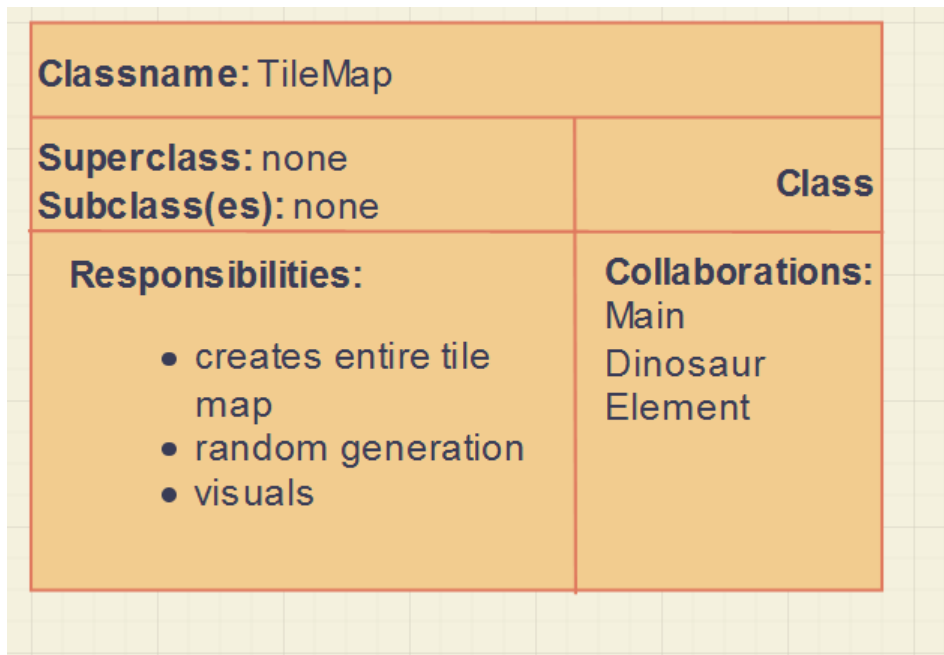
Rysunek 11: Klasa do eksportu do CSV.



Rysunek 12: Główna klasa.



Rysunek 13: Klasa do utworzenia GUI.



Rysunek 14: Klasa reprezentująca główną mapę kafelkową.

5 Diagramy Klas













Zaznaczając jeszcze polimorfizm jest zastosowany nie tylko jako metoda w Canivore, Omnivore i Hebivore - mowa o `getSpecieName()`, stosuje także polimorfizm, już w głównym kodzie w klasie Main, tam mam ustawione przykładowe działania:

- `dino instanceof Carnivore` - wtedy porusza się o 3 pola
- `dino instanceof Omnivore` - wtedy porusza się o 2 pola
- etc...

Poniżej diagramy klas, okazały się być bardzo rozbudowane, więc możliwe, że są słabo widoczne na niektórych wyświetlaczach, dlatego w archiwum głównym *ostrowski.zip* można je znaleźć pod *wszystkie_klasy.png*
























JurassicPark.Gui

Gui

- ☐ labelIterations: JLabel
- ☐ sliderIterations: JSlider
-
- ☐ labelSize: JLabel
- ☐ sliderSize: JSlider
-
- ☐ labelElements: JLabel
- ☐ sliderElements: JSlider
-
- ☐ labelSpecies: JLabel
- ☐ sliderSpecies: JSlider
-
- ☐ labelGrid: JLabel
- ☐ sliderGrid: JSlider
-
- ☐ labelThread: JLabel
- ☐ textThread: JTextField
-
- ☐ start: JButton
- ☐ set: JButton
- ☐ stop: JButton
- ☐ apply: JButton
- ☐ csv: JButton
-
-  Gui(): JPanel
-
-  getIterations(): int
-  getMapSize(): int
-  getSpecies(): int
-  getElements(): int
-  getGrid(): int
-  getDelay(): int
-
-  startReturn(): JButton
-  stopReturn(): JButton
-  setReturn(): JButton
-  applyReturn(): JButton
-  csvReturn(): JButton

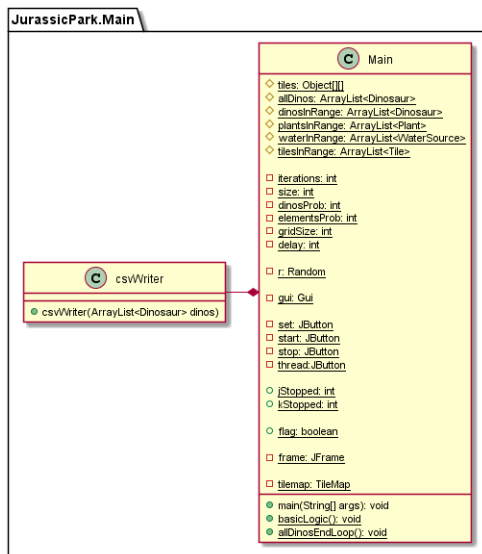
JurassicPark.map

TileMap

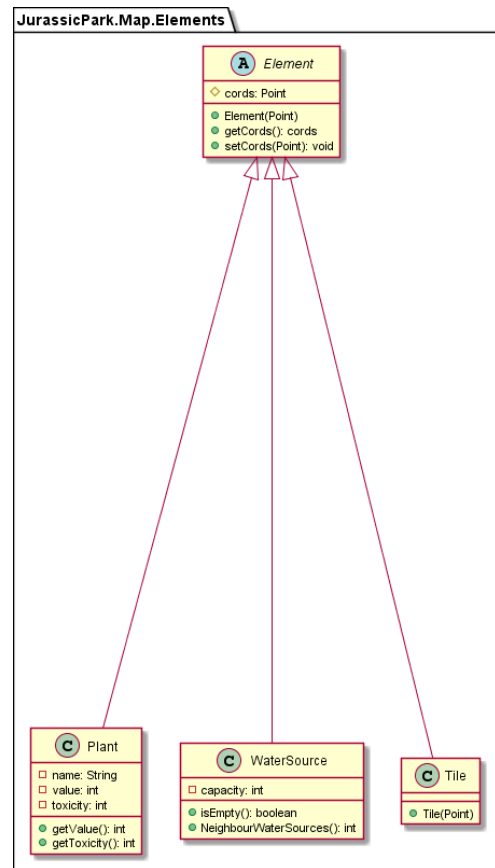
-  cCarnivore: Color
-  cOmnivore: Color
-  cHerbivore: Color
-  cPlant: Color
-  cWater: Color
-  cTile: Color
-
-  carnivore: Carnivore
-  omnivore: Omnivore
-  herbivore: Herbivore
-  plant: Plant
-  water: WaterSource
-  tile: Tile
-  gather: ArrayList<Object>
-
-  array: Object[]
-
- ☐ size: int
-
- ☐ gridSize: int
-
- ☐ dinosProb: int
- ☐ elementsProb: int
-
-  tiles: Object[][]
-
-  jsonHandler: JSOToSpecie
-  arraySpecie: ArrayList<Specie>
-  arrayDinos: ArrayList<Dinosaur>
-  arrayCarnivore: ArrayList<Specie>
-  arrayOmnivore: ArrayList<Specie>
-  arrayHerbivore: ArrayList<Specie>
-
-  TileMap(int,int,int)
-  paintComponent(Graphics): void

Rysunek 16: Układ pakietu Map.

Rysunek 15: Układ pakietu Main.

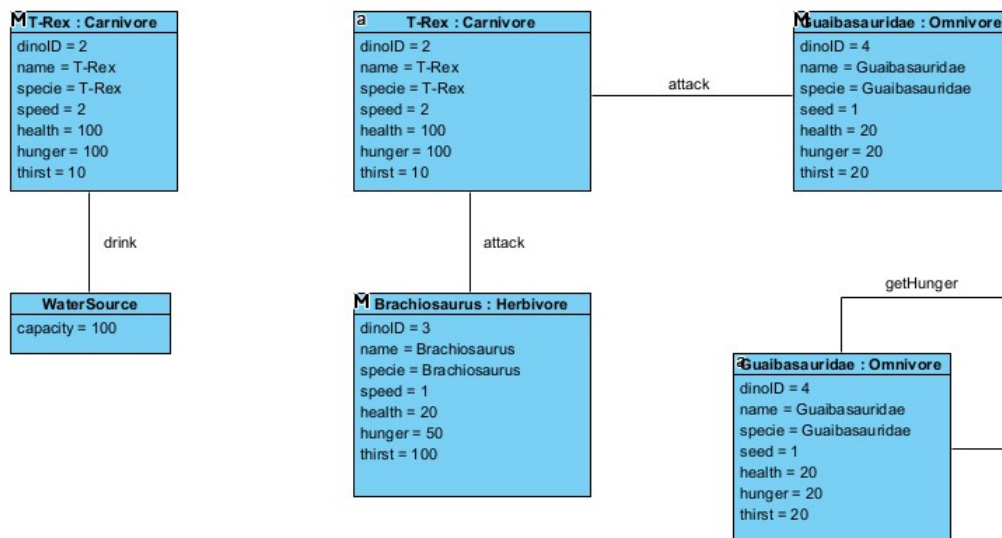


Rysunek 17: Układ pakietu Gui.

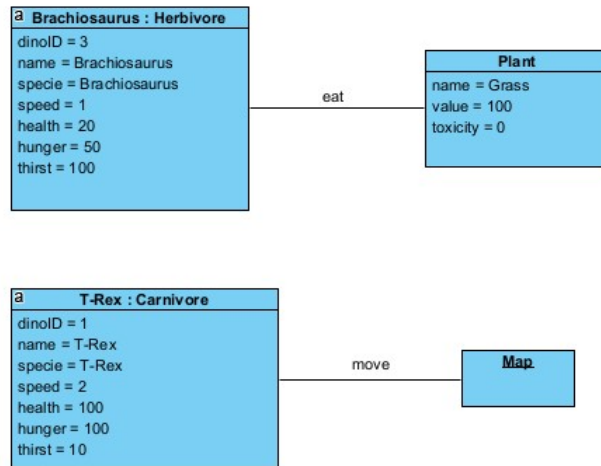


Rysunek 18: Układ pakietu Map.Elements.

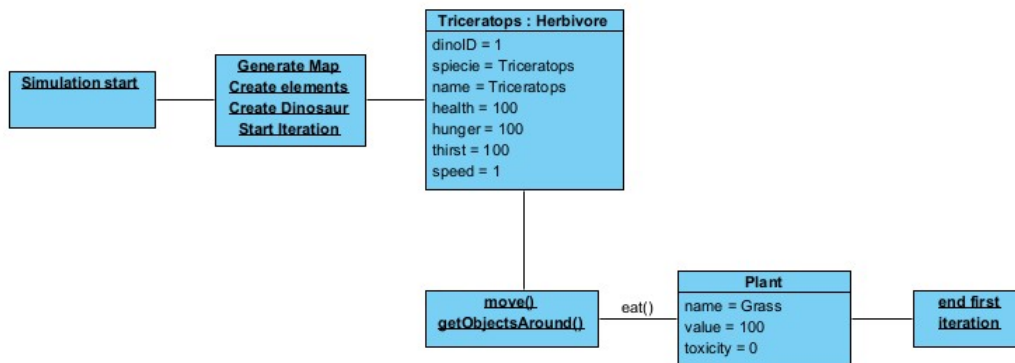
6 Diagramy obiektów



Rysunek 19: Diagram obiektów - przykład losowych akcji podczas jednej iteracji (mało szczegółowe)

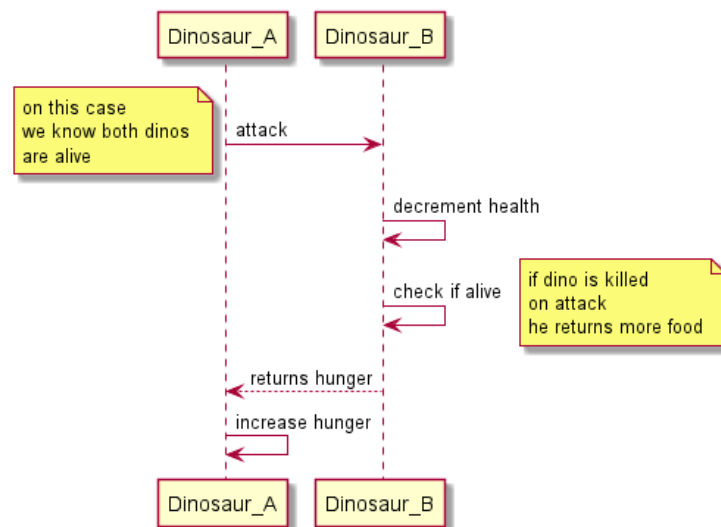


Rysunek 20: Diagramy obiektów - dalsze przykłady losowych akcji.

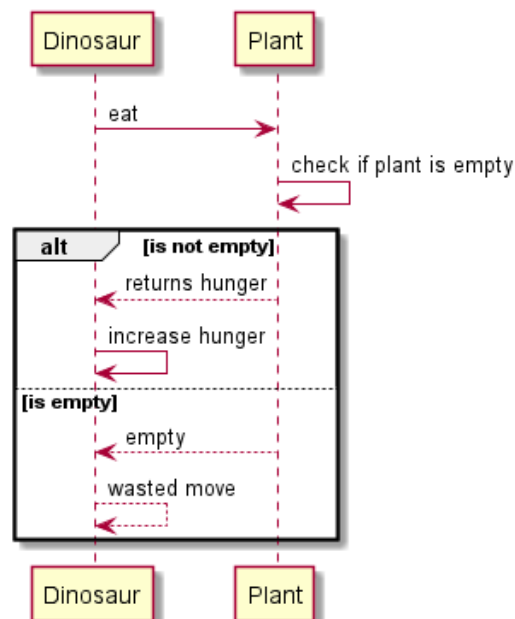


Rysunek 21: Diagram obiektów - prezentujący poglądowo pierwszą startową iterację.

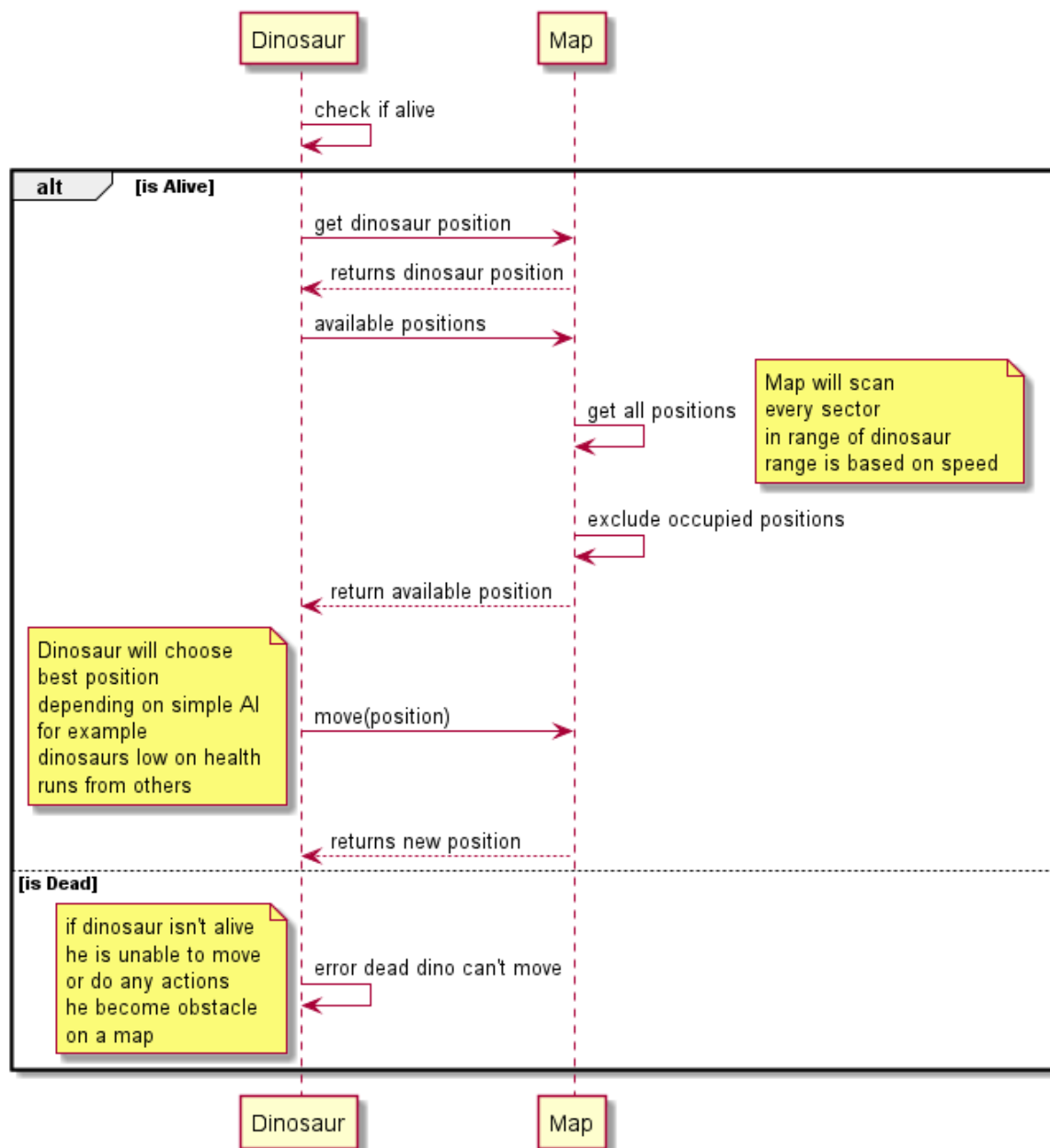
7 Diagram/diagramy sekwencji



Rysunek 22: Diagram prezentujący sekwencje ataku.



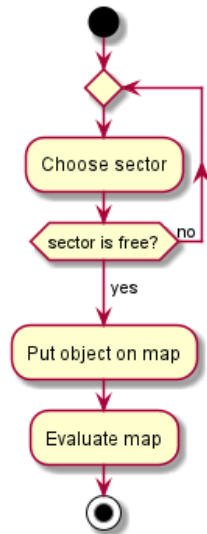
Rysunek 23: Diagram prezentujący sekwencje pożywiania się roślinami.



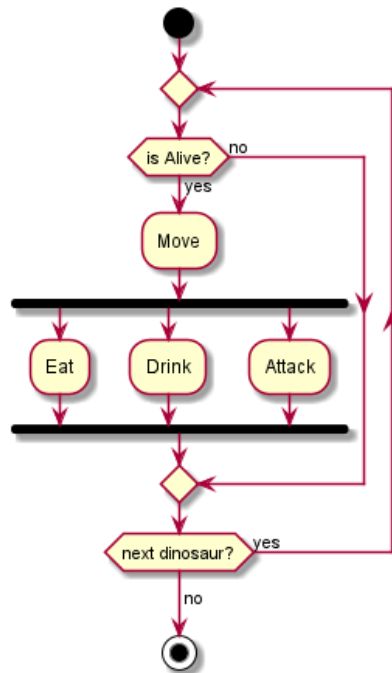
Rysunek 24: Diagram prezentujący sekwencje poruszania się dinozaura.

8 Diagram/diagramy aktywności

Uznałem tutaj, że aktywnością może też być generowanie obiektów na mapie w całej symulacji. Wszelkie aktywności odwołują się bardzo ogólnie do stworzonych (do poprzedniego zadania) klas.

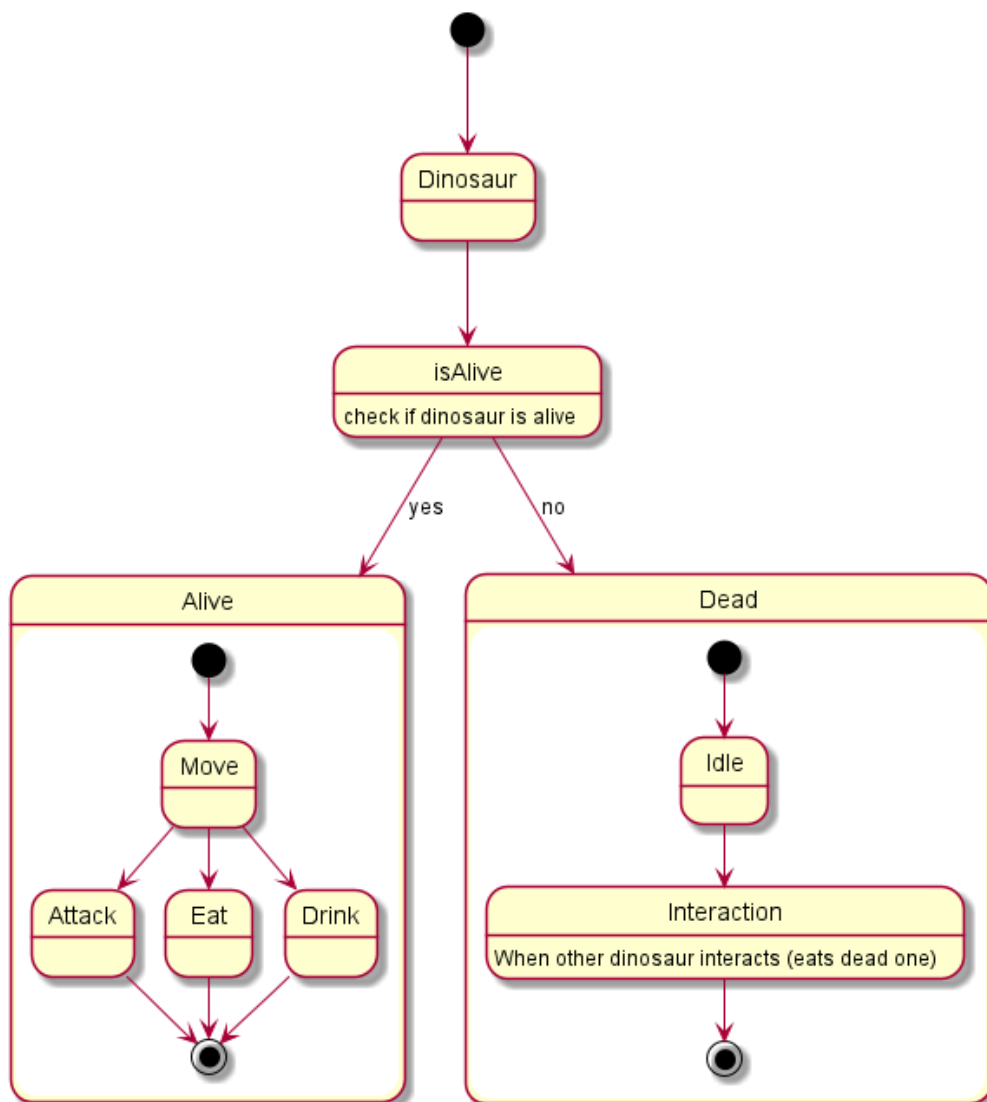


Rysunek 25: Diagram prezentujący generowanie obiektów na mapie.

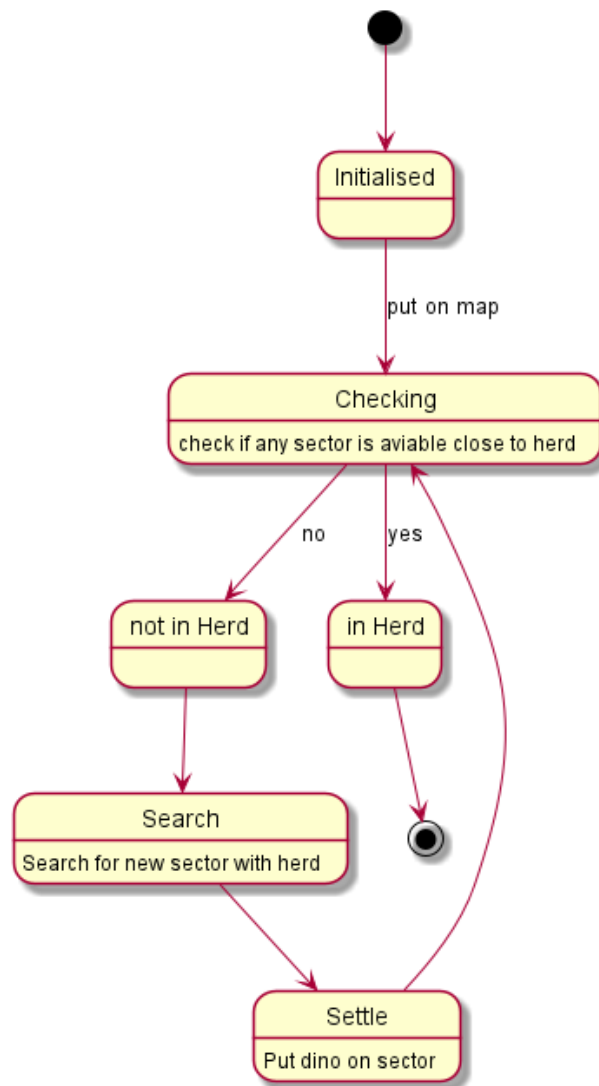


Rysunek 26: Diagram prezentujący możliwe aktywności dinozaura w iteracjach.

9 Diagram/diagramy maszyny stanów



Rysunek 27: Diagram prezentujący stany życia dinozaura i określone dla nich akcje.



Rysunek 28: Diagram prezentujący stany czy dany dinozaur jest w stadzie czy nie.

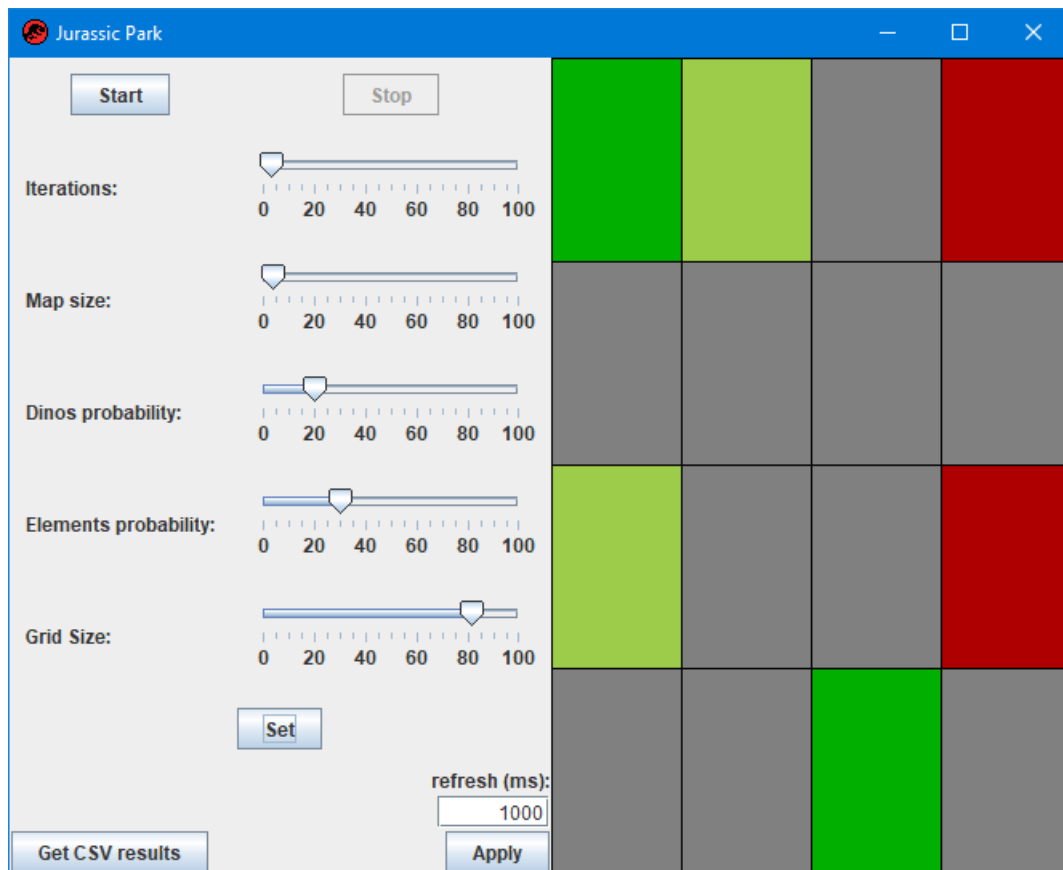
10 Dokumentacja

Pełną dokumentację można znaleźć w folderze *JurassicPark/doc* - wygenerowana w pełni javadoc'em, na podstawie komentarzy z kodu programu.

11 Dodatkowe elementy

11.1 GUI

Dla wszelkich nieścisłości przygotowałem krótki opis działania GUI:



Rysunek 29: Zdjęcie podglądowe GUI

- start - jak sama nazwa wskazuje, startuje symulację (thanks captain obvious) - przycisk startu można kliknąć tylko wtedy kiedy symulacja nie jest w trakcie, oraz przycisk start blokuje "set", lecz nie blokuje sliderów.
- stop - przycisk stopu działa tylko gdy symulacja jest uruchomiona
- set - możemy w każdej chwili tym przyciskiem wygenerować nową planszę, zgarnia on wszystkie dane z suwaków wyżej (powinienem był przygotować jeszcze przycisk restartu, żeby nie losowało nowej symulacji, ale zabrakło czasu)
- get csv results - tworzy i zapisuje wszystkie dinozaury do pliku *Simulation_Results.csv*
- apply - służy do zmieniania opóźnienia w wykonywaniu się głównej pętli, domyślna wartość 1000ms, czyli po 1 sekundzie, przeskakujemy na kolejne pole.
- suwak iterations - dobieramy ile razy ma wykonać się pętla sprawdzająca wszystkie pola na mapie (lepszym rozwiązaniem byłoby pole do wprowadzania liczby, jak na "refresh - apply", ale wtedy okienko gorzej wygląda)

- suwak map size - wielkość planszy S^2
- suwak dinos probability i elements probability - brak blokady na ustawienie 100% występowania obu zestawów elementów, co może skutkować dziwnymi generowaniami, polecam trzymać się w okolicach danych jak na obrazku
- grid size - rozmiar każdego pola

11.2 Odczytywanie z pliku JSON dinozaurów

Odczytywanie plików JSON, wykorzystałem do tego bibliotekę Google GSON, plik odczytywany to: *JurassicPark/src/main/resources/dinosaurs.json*

```
{
  "dinosaurs": [
    {
      "herdID": 1,
      "name": "Tyrannosaurus",
      "type": 0,
      "health": 100,
      "attack": 10,
      "hunger": 500,
      "thirst": 500
    },
    {
      "herdID": 2,
      "name": "Brachiosaurus",
      "type": 2,
      "health": 100,
      "attack": 0,
      "hunger": 500,
      "thirst": 500
    },
    {
      "herdID": 3,
      "name": "Triceratops",
      "type": 1,
      "health": 100,
      "attack": 10,
      "hunger": 250,
      "thirst": 250
    }
  ]
}
```

Rysunek 30: podgląd pliku dinosaurs.json

Zgodnie ze schematem ustawiamy po kolei:

- herdID - służy do identyfikowania konkretnych gatunków, liczba całkowita
- name - nazwa, może się powtarzać
- type - od 0 do 2, gdzie 0 to Carnivore, 1 to Omnivore i 2 to Herbivore
- health - liczba całkowita
- attack - liczba całkowita
- hunger - liczba całkowita
- thirst - liczba całkowita

Ostrzegam, trzeba się trzymać zasad wprowadzania, nie stworzyłem kontroli błędów, więc jeśli *type* podamy inny niż 0, 1 lub 2, to możliwe, że program przestanie działać.

11.3 Eksport do CSV

Zapisywanie wyników symulacji do pliku CSV o nazwie *Simulation_Results.csv*, zaimplementowałem tylko zapisywanie wszystkich dinozaurów i ich statystyk.

	A	B	C	D	E	F
1	dinoID	Specie	Name	Health	Hunger	Thirst
2	1	Brachiosaurus	Herbivore	90	480	440
3	2	Brachiosaurus	Herbivore	80	460	440
4	3	Kubozaur	Carnivore	100	200	190
5	4	Tyrannosaurus	Carnivore	80	450	460
6	5	Tyrannosaurus	Carnivore	100	460	440
7	6	Kubozaur	Carnivore	90	210	190
8	7	Brachiosaurus	Herbivore	90	480	440
9	8	Chalaceraptor	Omnivore	100	210	210
10	9	Lelozaur	Herbivore	195	960	960
11	10	Lelozaur	Herbivore	170	960	960
12	11	Tyrannosaurus	Carnivore	100	450	440
13	12	Brachiosaurus	Herbivore	100	440	480
14	13	Chalaceraptor	Omnivore	90	200	190
15	14	Lelozaur	Herbivore	180	960	940
16	15	Brachiosaurus	Herbivore	90	440	440
17	16	Chalaceraptor	Omnivore	95	200	190
18	17	Tyrannosaurus	Carnivore	100	460	440
19	18	Lelozaur	Herbivore	175	940	940
20	19	Brachiosaurus	Herbivore	100	440	480
21	20	Chalaceraptor	Omnivore	100	200	190
22	21	Tyrannosaurus	Carnivore	100	450	460
23	22	Lelozaur	Herbivore	180	940	940

Rysunek 31: Podglądowy wygląd wyeksportowanego pliku.