Programowanie Obiektowe – Projekt

Dokumentacja

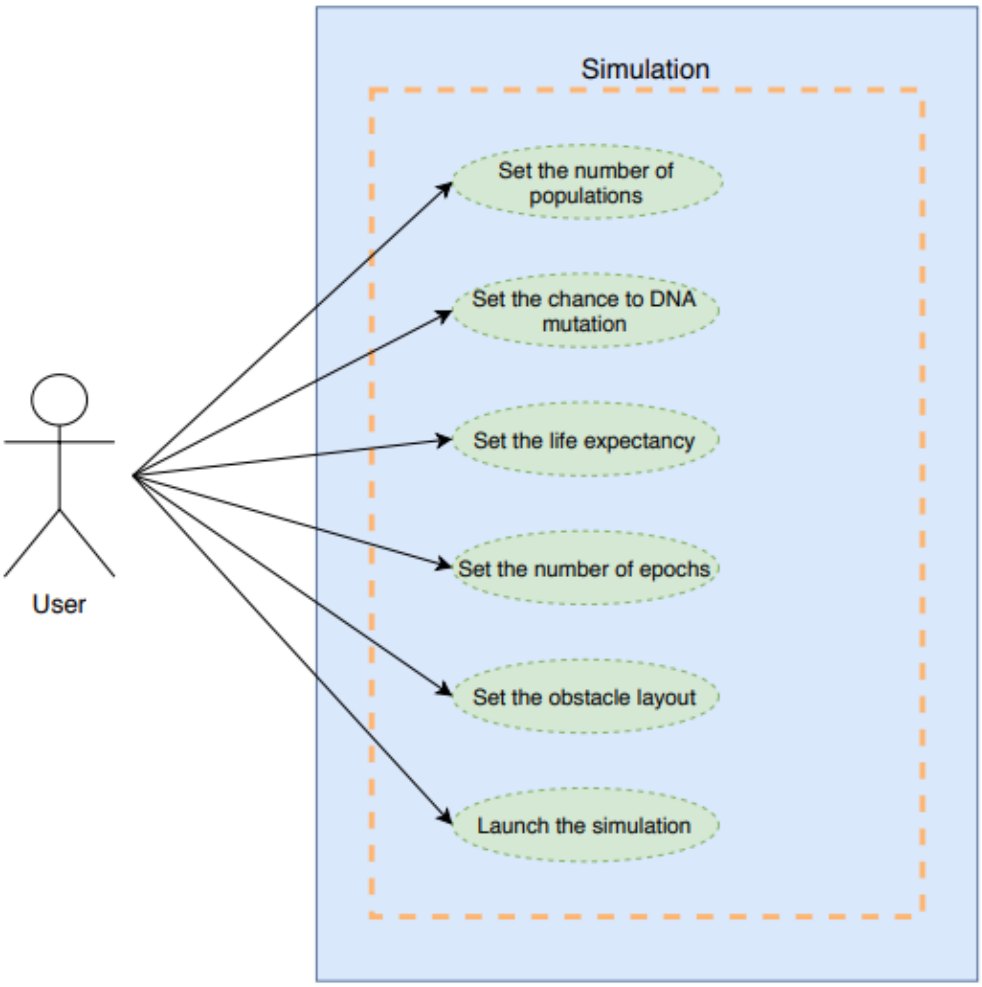
Skład grupy projektowej:

- Władysław Nowak, nr albumu: 252700 (lider grupy)
- Jakub Sroga, nr albumu: 252764

Opis zadania symulacji (z analizą czasownikowo-rzeczownikową):

Prosty program symulacyjny, przedstawiający uproszczony proces selekcji naturalnej z zastosowaniem prostego algorytmu genetycznego. Celem symulacji jest przeprowadzenie 2 różnie zachowujących się populacji przez przygotowaną wcześniej planszę na punkt końcowy. Na planszy umieszczone zostanie kilka przeszkód (osobnych obiektów klasy reprezentującej przeszkody), które mają zostać ominięte przez populacje. Każdy członek populacji jest oddzielnym obiektem klasy. Populacje będą różnić się od siebie o sposób poruszania się (będą miały różne prędkości średnie). Każdy obiekt tej klasy posiada własny kod genetyczny będący obiektem oddzielnej klasy zawierającej tablice wektorów opisujących jak ten obiekt ma się poruszać po planszy. "DNA" każdego członka populacji składane jest z kodu genetycznego jego rodziców, dodatkowo ma szansę na mutację czyli na zmianę losowego elementu tablicy na inną liczbę. Koniec każdej epoki symulacji oznacza stworzenie nowych obiektów będących potomstwem tych, którym najdalej udało się dojść w poprzedniej epoce. Dodatkowo możliwa na początku symulacji będzie zmiana liczności populacji początkowej, szansy na mutację, zmianę układu przeszkód oraz liczby epok do wykonania.

Diagram przypadków użycia:



Karty CRC dla klas:

Classname: Dna	
Superclass: none **Subclass**: none	
Responsibilities: - store an array of acceleration vectors - generate new array of acceleration vectors - mutate array of acceleration vectors - read next acceleration vector from an array - generate new Dna object from two already existing Dna objects	**Collaboration:**

Classname: Drawable (Interface)	
Superclass: none **Subclass**: Obstacle, PopulationMember	
Responsibilities: - imeplement function responsible for drawing	**Collaboration:**

Classname: DrawTask	
Superclass: none **Subclass**: Obstacle, PopulationMember	
Responsibilities: - defines simulation task to be repeated - draws population and obstacles on the screen	**Collaboration:** - Population - Obstacle

Classname: Main	
Superclass: javafx.application.Application **Subclass**: Obstacle, PopulationMember	
Responsibilities: - runs the simulation - specifies values of simulation parameters - configures JavaFX	**Collaboration:** - javafx.scene.canvas.Canvas

Classname: Movable (Abstract)	
Superclass: none **Subclass**: PopulationMember	
Responsibilities: - implements functions required for movement - implements position of an object - moves an object	**Collaboration:** - Vector

Classname: Obstacle	
Superclass: Drawable **Subclass**: none	
Responsibilities: - specifies position of obstacles in simulation - checks if given coordinates are a part of obstacle	**Collaboration:**

Classname: Population	
Superclass: none **Subclass**: none	
Responsibilities: - starts and ends new epochs of the simulation - stores population members	**Collaboration:** - FastPopulationMember - SlowPopulationMember

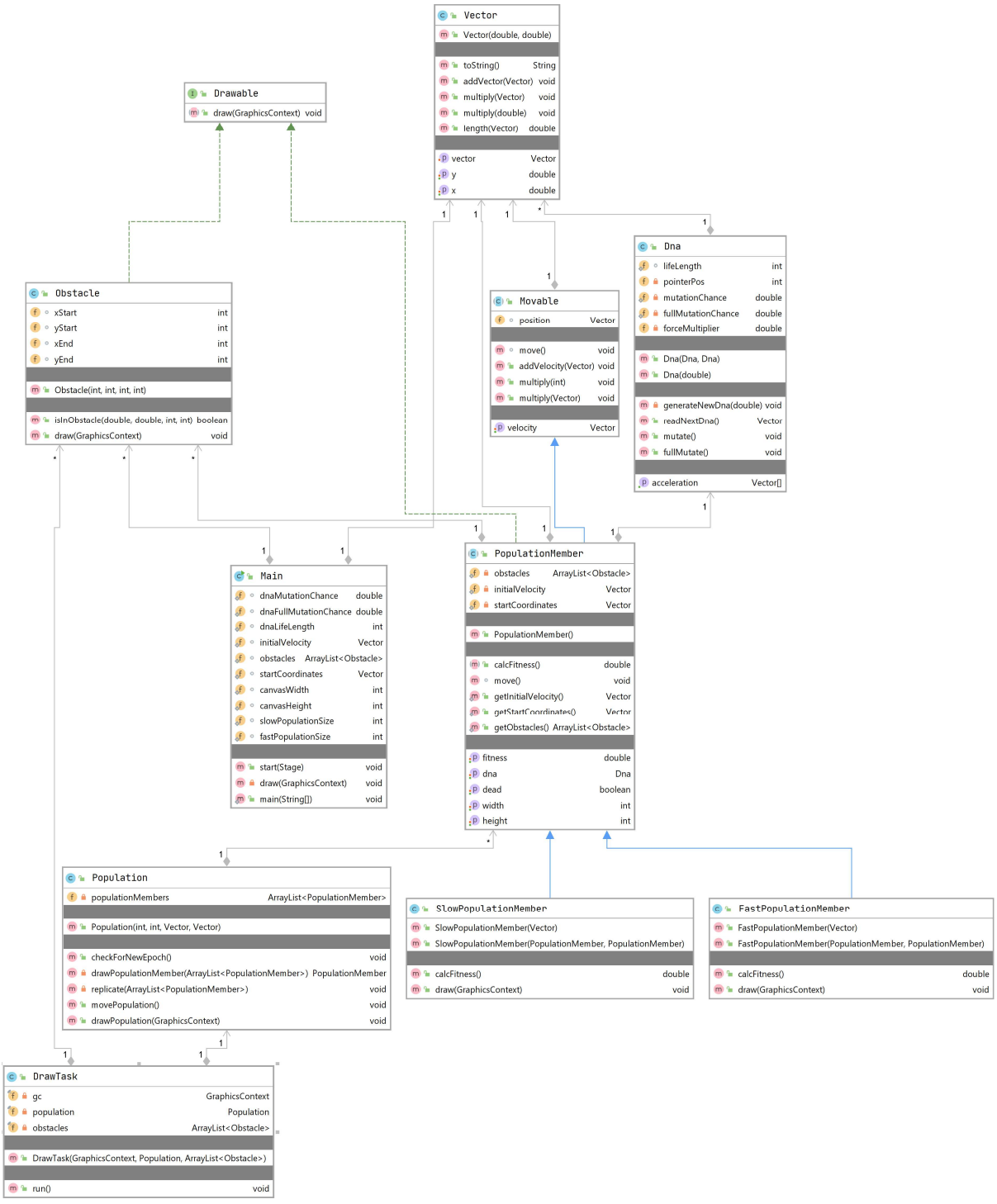
Classname: PopulationMember (Abstract)	
Superclass: none **Subclass**: FastPopulationMember, SlowPopulationMember	
Responsibilities: - defines basic properties and methods of specimens - store DNA of population members	**Collaboration:** - Dna

Classname: SlowPopulationMember	
Superclass: PopulationMember **Subclass**: none	
Responsibilities: - define velocity and size of slow population members - move slow population members - draw slow population members - calculate fitness	**Collaboration:**

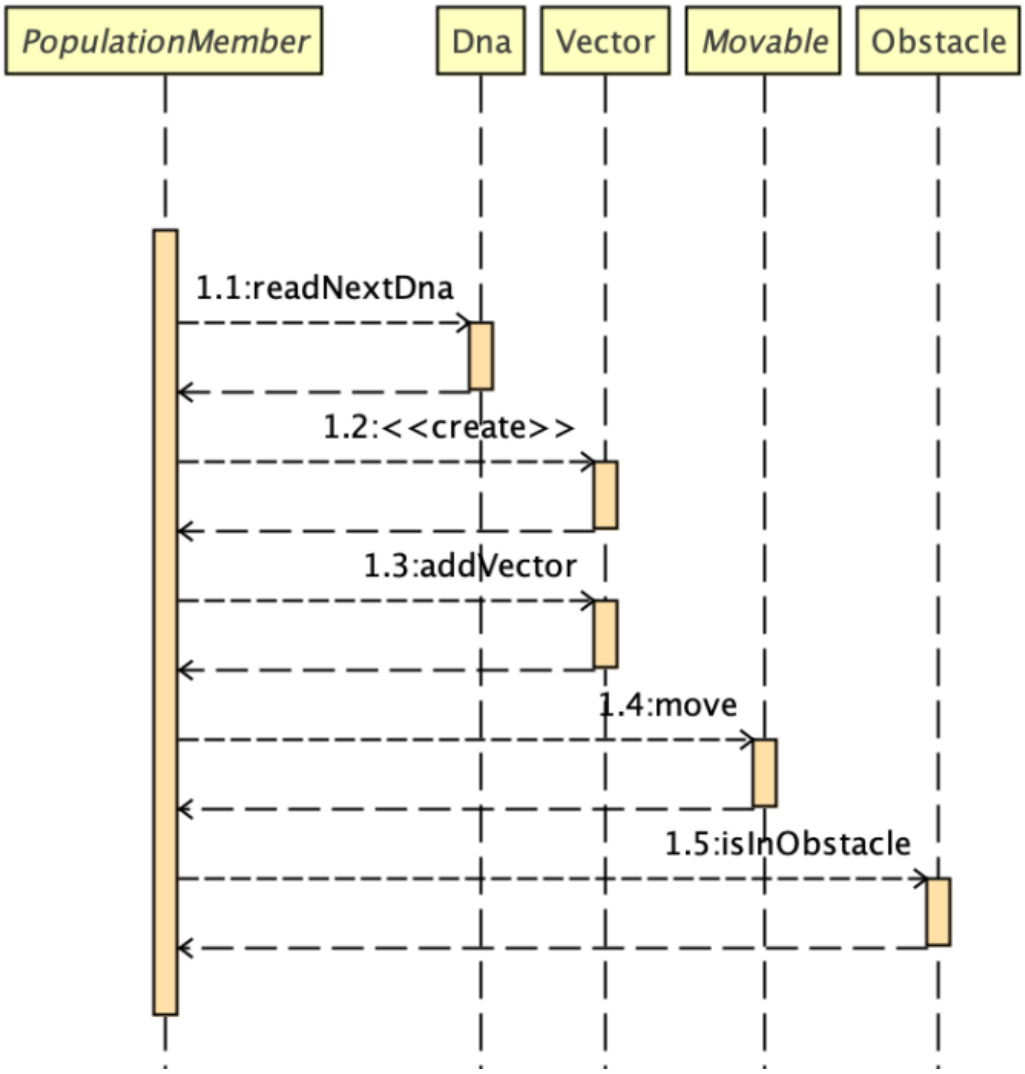
Classname: FastPopulationMember	
Superclass: PopulationMember **Subclass**: none	
Responsibilities: - define velocity and size of fast population members - move fast population members - draw fast population members - calculate fitness	**Collaboration:**

Classname: Vector	
Superclass: none **Subclass**: none	
Responsibilities: - store 2 dimensional vector - add two vectors - multiply vector by constant - multiply vector by other vector	**Collaboration:**

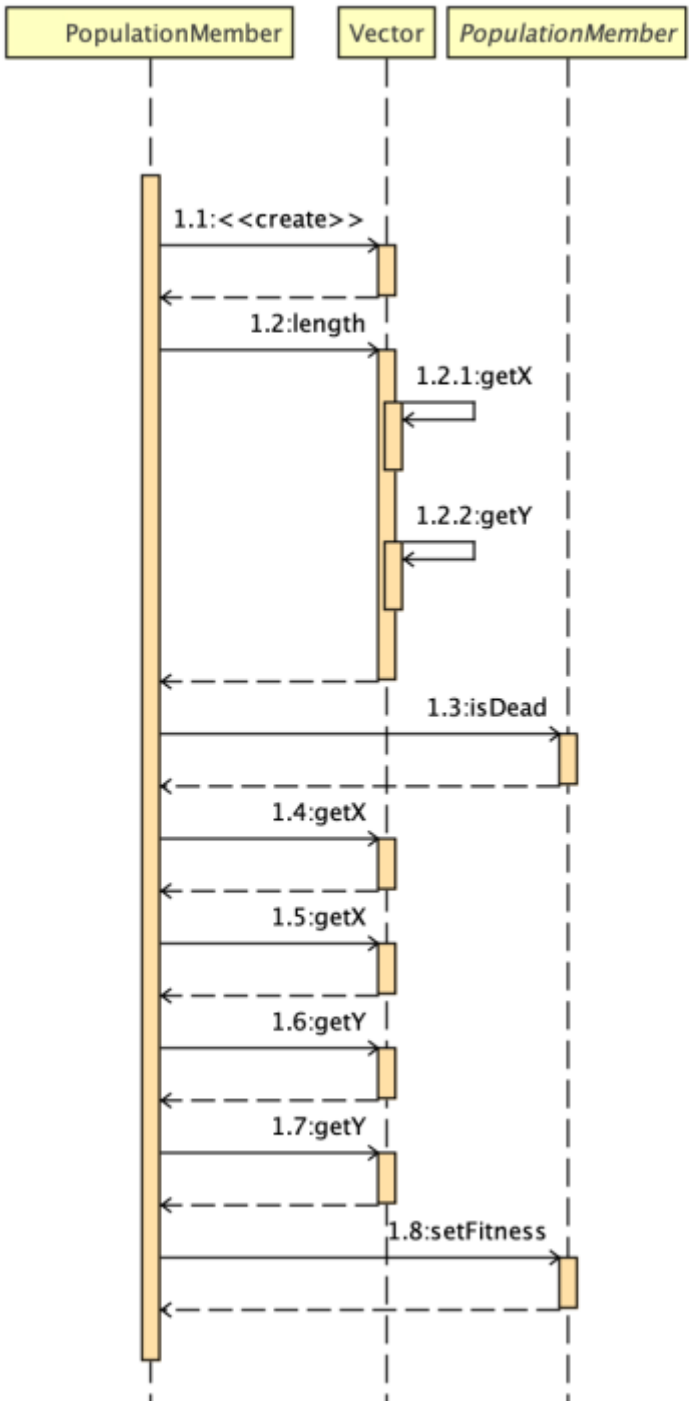
Diagram Klas



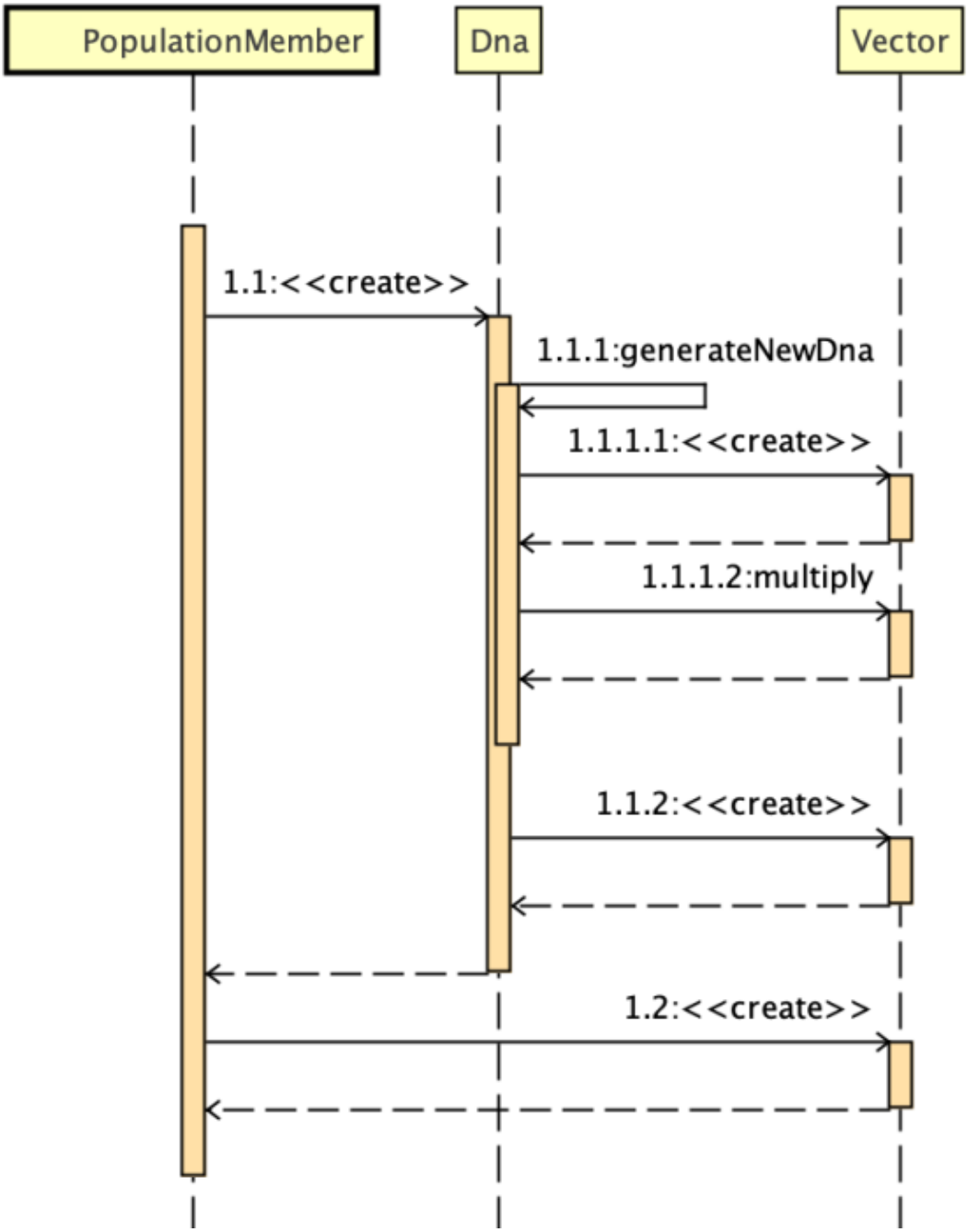
Diagramy sekwencji:



Poruszanie się osobników

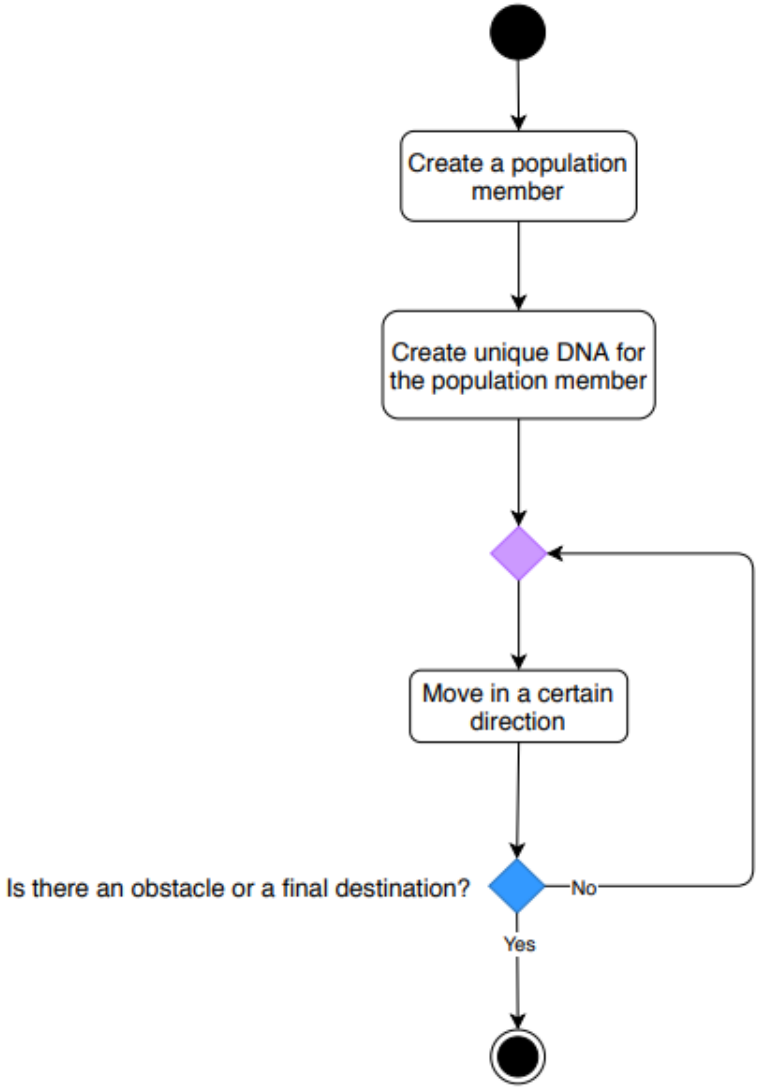


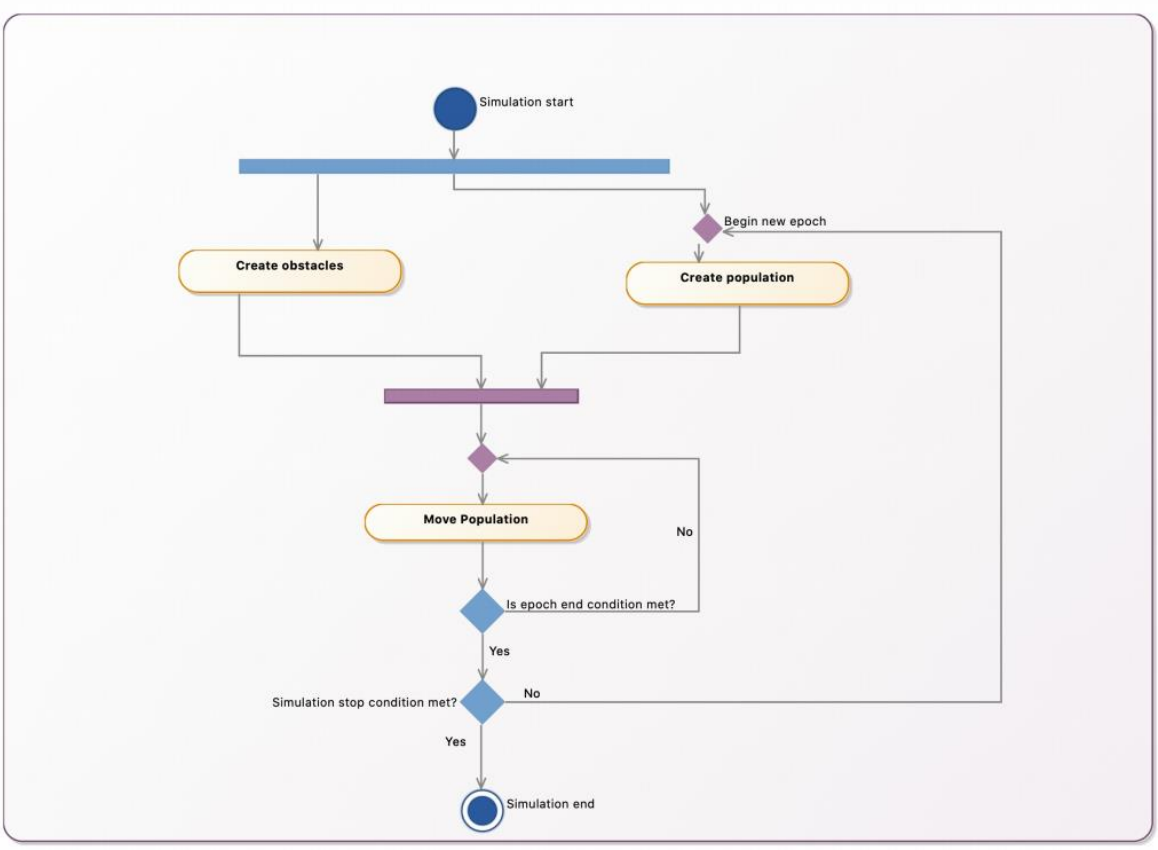
Liczenie wartości dopasowania (fitness)



Tworzenie osobników

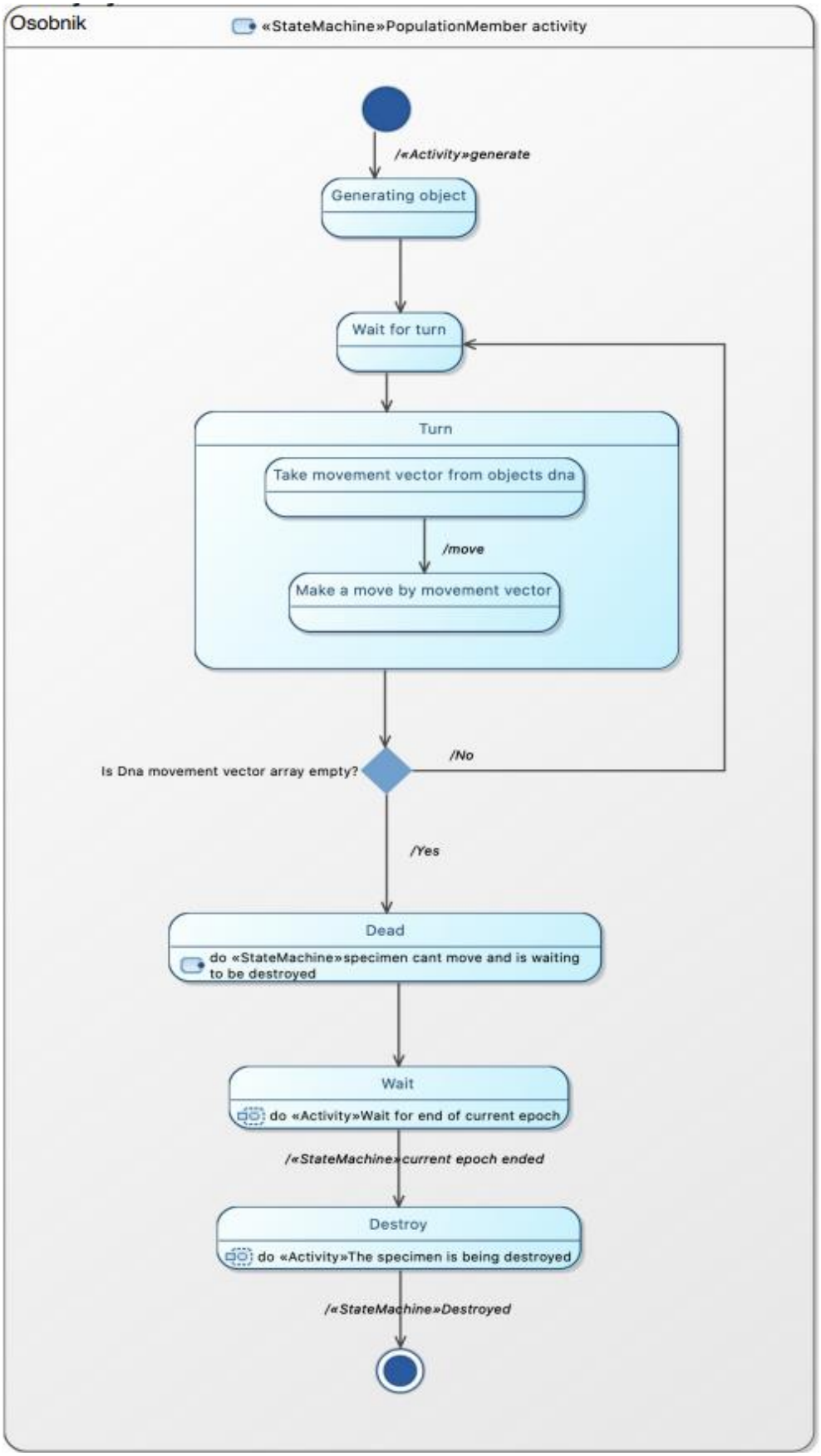
Diagramy aktywności:



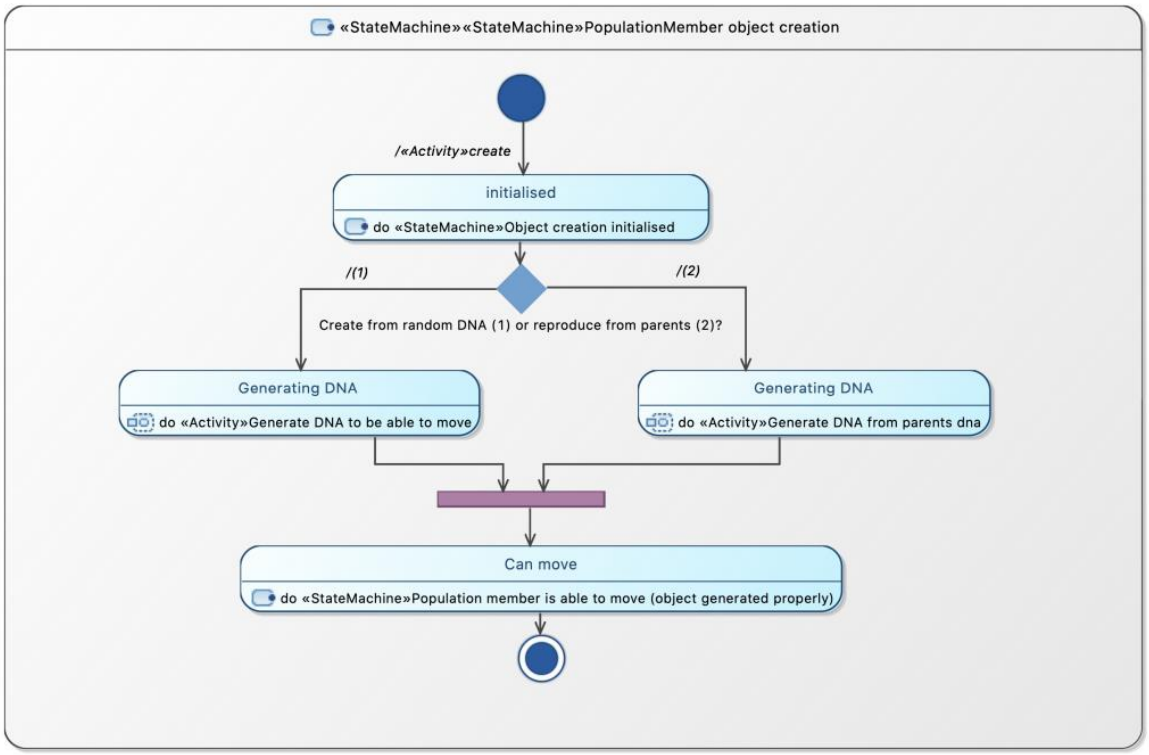


Przebieg symulacji

Diagramy maszyny stanów:



Osobnik



Tworzenie osobników

Dokumentacja wykonana na podstawie komentarzy w kodzie: https://ultux.github.io/GeneticAlgorithm/simulation/package-summary.html