Członkowie grupy: Mateusz Bienkiewicz

Język programowania: Java

Temat pierwszy: Las przy wiosce

Symulacja życia w lesie przy wiosce.
Zwierzęta: W lesie mogą mieszkać dziki, jelenie, wilki i lisy. Każdy z nich musi jeść, spać i się rozmnażać. Każde zwierzę żyjące w lesie zaczyna życie po porodzie jako młody, a po upłynięciu określonego dla każdego gatunku czasu staje się samcem lub samicą. Jelenie, dziki i wilki żyją w grupach. Zachodzą określone interakcje pomiędzy określonymi gatunkami. Zwierzęta są dodatkowo określone prze wiek, zwinność, zdolności waleczne, wyspanie, głód. Te cechy mogą wpływać na wynik spotkania z innymi żywymi istotami. Przykładem interakcji może być spotkanie się samicy i samca tego samego gatunku podczas okresu rozrodczego, w wyniku czego powstaje za pewien czas nowy obiekt będący młodym zwierzęciem. Jelenie i dziki żywią się krzewami w lesie i uprawami rolników na przyległych do wioski polach. Lisy polują jedynie na kury, które może mieć na podwórku każdy z mieszkańców wsi, jednak ponieważ każdy mieszkaniec wsi może zastawić na lisa śmiertelną pułapkę, lis musi mieć jeszcze określony współczynnik szczęścia, który jest większy im większa jest zwinność lisa. Wilki polują na pozostałe gatunki w lesie (częściej zdecydują się zaatakować na przykład łanię jelenia niż dzika). Zwierzęta szukają siebie nawzajem na podstawie znaków zostawionych na drzewach i wydawanych dźwięków. Każde zwierzę co pewien czas musi się rozglądać i nasłuchiwać.
Mieszkańcy wioski: Niektórzy wieśniacy mogą być myśliwymi, którzy co pewien czas wychodzą do lasu na polowanie na każdy z 4 gatunków leśnych zwierząt. Nie mogą polować na określone gatunki w miesiącach, kiedy jest to zabronione prawem. Myśliwy szuka zwierząt na podstawie tropów na ziemi i znaków na drzewach. Myśliwym przez natchnienie w pewnym momencie może stać się ktoś nowy. „Normalny” wieśniak również może pójść do lasu. W lesie są drogi. Każdy człowiek wchodzący do lasu rozpoczyna wędrówkę od drogi. Myśliwi niemal na pewno z niej zboczą, ale zwykli wieśniacy, jeśli już idą do lasu na spacer, to przeważnie trzymają się dróg. Ponieważ odwiedzający las mogą natknąć się na przykład na dzika, każdy człowiek również ma ustalony poziom zwinności, od którego zależy, czy ucieknie). Teren jest podzielony na części. Ludzie mogą się rodzić i umierać, Każdy ma określony wiek. Myśliwi mają dodatkowo określony współczynnik umiejętności łowieckich. Nowe obiekty mieszkańców wsi tworzą się na podstawie stałego współczynnika przyrostu naturalnego. Starzy mieszkańcy umierają. Dodatkową cechą ludzi jest ambicja_hodowcy. Zależy od niej do posiadania jak dużej ilości kur dąży człowiek. Jednak należy pamiętać, że kur nie może posiadać mieszkaniec wioski przed 20 rokiem życia. Taki mieszkaniec ma określoną ambicję hodowcy, lecz na nic ona jeszcze nie wpływa.
Teren: Teren jest podzielony na części. W zależności od pogody każda część gleby jest bardziej lub mniej wilgotna. Wpływa to na rozwój roślinności. Teren dzieli się na takie rodzaje: pola, las, teren zabudowany wsią, wolny teren. Pola mogą być nieużywane, zaorane, z zasadzonymi roślinami lub z rozwiniętymi roślinami. Ponieważ myśliwi chodzą na polowanie tylko do lasu, to tu tylko będzie rozważane, czy na kawałku ziemi jest trop zwierzęcia. Kawałki leśnego terenu mają określone zagęszczenie drzew oraz gęstość „krzaków”. Na terenach wolnych jest szansa na osiedlenie się nowych ludzi w wieku 20+lat. Każdy teren zabudowany wsią ma określoną ilość kur. Oczywiście lis chętniej zapoluje na kury będące na obrzeżach wsi niż na te w środku. Pogoda nie jest podzielona na strefy. W tej samej chwili na całej „mapie” panują te same warunki pogodowe.
Cel: Symulacja ma na celu wypisanie do pliku CSV ilości dzików, jeleni, wilków, lisów, kur, myśliwych i nie-myśliwych po minięciu określonego czasu przy losowych lub odczytanych z pliku wartościach początkowych.

Temat 2: Wężyki i bakterie
OPIS SŁOWNY:
Na mapie są bakterie i węże i pożywienie dla węży.

WSZELKIE WARTOŚCI LICZBOWE ZAPISANE CYFRAMI SĄ PRZYKŁADOWYMI WARTOŚCIAMI PODANYMI W PLIKU KONFIGURACYJNYM I. MOGĄ ZOSTAĆ W NIM ZMIENIONE POZA PROGRAMEM SYMULACYJNYM, KTÓRY NASTĘPNIE JE WCZYTA.

Długość i szerokość prostokątnej planszy oraz ilość węży i grup bakterii jest podana w pliku konfiguracyjnym. Grup bakterii nie może być więcej niż powoła sumy ilości pól planszy, a węży nie może być więcej niż ćwierć sumy ilości pól planszy.

Węże: Węże początkowo mają długość równą jednemu polu – mają tylko głowę. Węże nie widzą bakterii i prawie zawsze (chyba że są chore) dążą do najbliższego pakietu pożywienia. W pakiecie pożywienia zawsze znajduje się jedzenie normalne, które przedłuża węża o losową długość od jeden do trzech. Oprócz tego w pakiecie pożywienia może znajdować się jedzenie specjalne (każde trwające 5 tur): przyspieszające węża (przesuwa się o dwa pola na turę), zwalniające węża (przesuwa się o jedno pole na dwie tury) lub uodporniające na bakterie (bakterie nie mogą z takim wężem wejść w interakcję i jeśli się na niego natkną, to przez taka grupa jest ogłuszona). Każde z tych trzech specjalnych posiłków ma losowane prawdopodobieństwo wystąpienia w pakiecie pożywienia – może się pojawić w co 10 pakiecie. Jeśli wąż wejdzie na pole zajmowane przez ciało innego węża, to zostaje ogłuszony i nie może się ruszać. Wąż nie może się już wydłużać, jeśli po wydłużeniu byłby dłuższy niż długość lub szerokość planszy (mniejsza z tych dwóch wartości).

Ogłuszenie trwa 5 tur. Po ogłuszeniu wąż lub grupa bakterii kontynuuje wędrówkę.

Bakterie: Bakterie są małe, średnie lub duże. Małe stają się średnie po 10 turach, a średnie duże po 20 turach. Ich wielkość jest wyrażona przez liczbę przypisaną polu klasy. Bakterie łączą się w grupy, które poruszają się po mapie do najbliższej mniejszej grupy bakterii, aby z nią walczyć. Grupy bakterii walczą, wysyłając na pojedynek jeden na jednego bakterie od tych najmniejszych. Wysyłają na pojedynek tę samą bakterię, aż nie zginie. Wygrywa bakteria mająca większy losowany przy każdej walce współczynnik sukcesu (losowy od zera do stu) w walce przemnożony przez 1 dla małych bakterii, 2 – średnich, 3 - dużych, czyli przez liczby określające wielkość bakterii. Jeśli w przeciwnej drużynie bakterii zostanie mniej niż 25%, to te 25% dołącza się do wygrywającej drużyny. Bakterie, które po drodze „na walkę” natkną się na węża (lecz nie na jego głowę), to wchodzą na jego skórę i podróżują z nim, rozmnażając się (liczba każdego rodzaju jednostek w drużynie zostaje podwojona). Jeśli natkną się na pożywienie węży, to rozmnażają się na nim, również podwajając swą liczbę. Jeśli w tym czasie wąż zje jedzenie, to bakterie rozmnażają się dalej w żołądku, jeśli ta drużyna bakterii wygrała już od początku symulacji 3 walki, w przeciwnym razie bakterie giną. Jeśli w drużynie bakterii jest 50 lub więcej bakterii, to wąż po zakończeniu rozmnażania bakterii ginie, a bakterie wychodzą tam, gdzie był jego przód (jego głowa). Jeśli bakterii jest mniej niż 50, to po rozmnażaniu bakterie wychodzą jedno pole za końcem węża. Jeśli drużyna bakterii od początku symulacji wygrała ponad 6 walk i wąż natknie się głową na taką grupę, to bakterie rozmnażają się w nim 8 tur, po czym wąż ginie, jeśli jest ich 100 lub więcej, a jeśli jest ich mniej to wąż przetrwa, a bakterie po rozmnażaniu wyjdą jedno pole za końcem węża. Jeśli wąż natknie się głową na „mniej doświadczoną w walce” grupę, to niszczy ją.
Każde rozmnażanie bakterii trwa 8 tur.
Początkowy skład grup bakterii (ilość małych, średnich, dużych) jest losowany. Bakterii każdej wielkości może być w grupie od 0 do 10.

Antidotum: Wąż może z każdego zarażenia bakteriami wyleczyć się, zjadając antidotum, które niszczy wszelkie pasożytujące na nim (lub w nim) bakterie. Jakkolwiek zarażony wąż zamiast do jedzenia dąży do antidotum.

Pożywienia na mapie zawsze jest tyle co węży, antidotum pojawia się jedno co 7 tur i po 10 turach znika.

Symulacja losuje początkowe położenie węży, ilość początkową bakterii i położenie ich grup, oraz lokalizacje pakietów pożywienia/antidotum podczas symulacji. Wąż, grupa bakterii, pakiety pożywienia i antidotum nie mogą mieć wylosowanej pozycji „stworzenia się" na polu zajmowanym już przez coś innego. Po określonej w pliku konfiguracyjnym liczbie tur ilości węży (i ich długości) oraz grup bakterii (i ich składy) są zapisywane do pliku CSV.

Analiza rzeczownikowo-czasownikowa:

Symulacja wykonuje się przez określoną w pliku konfiguracyjnym ilość tur. Wczytywacz danych wczytuje dane z pliku konfiguracyjnego, w którym są zapisane jako tekst, i umie z tekstu wyjmować liczbę będącą jego ostatnim wyrazem. W symulacji są wolne pola, które mogą podawać swoje współrzędne. Mapa symulacji to spoty, na których mogą być przechowywane obiekty i hashmapy udostępnianych przez nie informacji. Obiekty biorące udział w symulacji mogą zwrócić swoje współrzędne, ustawić sobie nazwę i ją zwrócić, zebrać i zwrócić informacje o sobie, zwrócić informację, czy nadal żywe, ustawić się na martwych, usunąć się ze spotów i biorących udział w symulacji (zginąć), wylosować sobie liczbę z danego przedziału, wylosować sobie spot, na którym mają się stworzyć, oraz zareagować na otrzymane informacje, lecz najważniejsza z przeprowadzanych przez nie operacji to przeprowadzanie swojej tury. Żywe obiekty są obiektami, które poruszają ze spota na spot i wchodzą w interakcje z obiektami będącymi na tym samym spocie, z którego mogą usuwać informacje o sobie, gdy to potrzebne. Antidotum jest zwykłym obiektem, który ginie w swojej turze po określonej ilości tur istnienia lub po zjedzeniu przez węża lub grupę bakterii. Jedzenie natomiast jest zwykłym obiektem, który może się zginąć w swojej turze tylko po byciu zjedzony przez węża. Wąż jest żywym obiektem, który może w swojej turze wejść na spota zostać zarażonym przez grupę bakterii z jedzenia lub nie z jedzenia, wyleczyć się z zarażenia zjadając antidotum, zjeść jedzenie, w wyniku czego przedłużyć się, zwolnić, przyspieszyć lub zdobyć ochronę na bycie zarażonym. Jeśli wąż miałby wejść na spot z innym wężem, to zostaje na starym spocie i ogłusza się. Zarażony wąż szuka celu będącego najbliższym antidotum i rusza się do niego. Bakterie mogą zwiększać swój level i ilość przeżytych tur. Są one częścią grup bakterii, które są żywymi obiektami. Grupa bakterii w swojej turze: jeśli wejdzie na węża lub jedzenie, to może podwoić swoją liczebność, jeśli wejdą na antidotum lub zostaną zjedzone przez węża, na którym nie mogą podwoić swojej liczebności z powodu za małej liczby wygranych walk, to giną, jeśli wejdą na inną grupę bakterii, to walczą z nią i wygrana grupa powiększą swoje bakterie i liczebność o pozostałą po walce część drugiej grupy, a przegrana grupa ginie. Zapisywacz zapisuje końcowe wyniki symulacji mówiące o tym, jak liczne grupy bakterii i jak długie węże zostały, do pliku zewnętrznego.

Parametry symulacji ustalane w pliku konfiguracyjnym:

- ILOSC TUR SYMULACJI
- Dlugosc planszy
- Szerokosc planszy
- Czas dzialania jedzenia specjalnego
- Rzadkosc wystapienia jedzenia specjalnego
- Czas ogluszenia
- Czas ewolucji malej bakterii na srednia
- Czas ewolucji sredniej bakterii na duza

- Minimalna liczba poczatkowa malych bakterii w grupie bakterii
- Minimalna liczba poczatkowa srednich bakterii w grupie bakterii
- Minimalna liczba poczatkowa duzych bakterii w grupie bakterii
- Grupa bakterii na pakiecie jedzenia nie ginie po zjedzeniu przez weza po tylu wygranych
- Waz ginie po rozmnozeniu sie w nim tak licznej grupy bakterii z pakietu jedzenia
- Grupa bakterii nie ginie po zjedzeniu przez weza po tylu wygranych
- Waz ginie po rozmnozeniu sie w nim tak licznej grupy bakterii
- Czas rozmnazania sie grupy bakterii
- Czas miedzy pojawieniem sie antidotum
- Czas istnienia antidotum
- Tyle procent bakterii musi zostac w grupie, zeby zostala wchlonieta w sklad drugiej grupy
- Maksymalna liczba poczatkowa malych bakterii w grupie bakterii
- Maksymalna liczba poczatkowa srednich bakterii w grupie bakterii
- Maksymalna liczba poczatkowa duzych bakterii w grupie bakterii
- Ilosc wezy
- Ilosc grup bakterii

Karty CRC:

Classname: Antidotum	
Superclass: Obiekt Subclass(es): brak	
Responsibilities: istnienie antidotum: • stwarzanie • wykonywanie tury • reagowanie na otrzymane informacje • zbieranie informacji o sobie • ustaw czas istnienia antidotum	Collaboration: • informacje

Classname: Bakteria	
Superclass: brak Subclass(es): brak	
Responsibilities: istnienie bakterii: • stwarzanie • zwiększanie poziomu • zwiększanie ilości przeżytych tur • zwracanie poziomu • zwracanie ilości przeżytych tur	Collaboration: brak

Classname: Grupa	
Superclass: Zywe Subclass(es): brak	
Responsibilities: istnienie grupy: • stwarzanie • wykonywanie tury • reagowanie na otrzymane informacje • zbieranie informacji o sobie	Collaboration: • BakteriaI • WspI

• walka z inną grupą • wywoływanie zwiększania ilości przeżytych tur bakterii • usuwanie swoich informacji ze spota • zwracanie tablicy z liczbami bakterii o poszczególnych levelach w grupie • ustaw czas rośnięcia z levela 1 do 2 • ustaw czas rośnięcia z levela 2 do 3 • ustaw od jakiego procenta pozostałej liczby członków można pochłoną drugą grupę w walce • ustaw minimalną liczbę bakterii 1 levela, jaka może się stworzyć przy tworzeniu • ustaw maksymalną liczbę bakterii 1 levela, jaka może się stworzyć przy tworzeniu • ustaw minimalną liczbę bakterii 2 levela, jaka może się stworzyć przy tworzeniu • ustaw maksymalną liczbę bakterii 2 levela, jaka może się stworzyć przy tworzeniu • ustaw minimalną liczbę bakterii 3 levela, jaka może się stworzyć przy tworzeniu • ustaw maksymalną liczbę bakterii 3 levela, jaka może się stworzyć przy tworzeniu	

Classname: Main	
Superclass: brak Subclass(es): brak	
Responsibilities: przeprowadzenie symulacji i obsługa jej ważnych elementów: • zwróć listę obiektów biorących udział w symulacji • ustaw liczbę tur, co jaką stwarza się nowe antidotum • zwróć spot o konkretnych współrzędnych • zwróć puste pole o konkretnym indeksie w liście pustych pól • zwróć listę pustych pól • usuń puste pole o konkretnym indeksie z listy pustych pól • usuń puste pole o konkretnych współrzędnych z list pustych pól • ustaw ilość węży i jedzenia do stworzenia • ustaw ilość grup bakterii do stworzeniami • ustaw długość „planszy”	Collaboration: • ExistingI • WspI

• ustaw szerokość „planszy” • ustaw ilość „tur” • symulacja	

Classname: Obiekt	
Superclass: brak Subclass(es): Zywe, Antidotum, Pakiet_jedzenia	
Responsibilities: przeprowadzenie operacji wspólnych dla obiektów: • losowanie spota, na którym ma się stworzyć • losowanie liczby całkowitej z przedziału • usuwanie z symulacji • ustawianie jako martwego • zwracanie współrzędnych swoich • ustawianie swojej nazwy • zwróć zmienną logiczną o tym, czy żyje • zwróć hashmapę informacji o obiekcie • zwróć nazwę obiektu	Collaboration: • informacje

Classname: Pakiet_Jedzenia	
Superclass: Obiekt Subclass(es): brak	
Responsibilities: istnienie antidotum: • stwarzanie • wykonywanie tury • reagowanie na otrzymane informacje • zbieranie informacji o sobie • ustaw prawdopodobieństwo wystąpienia w jedzeniu mocy specjalnej	Collaboration: • informacje

Classname: ReadData	
Superclass: brak Subclass(es): brak	
Responsibilities: odczytywanie danych z plik konfiguracyjnego: • odczytaj dane z pliku i zapisz je. • odczytaj liczbę z ostatniego wyrazu linijki tekstu	Collaboration: brak

Classname: Waz	
Superclass: Zywe Subclass(es): brak	
Responsibilities: istnienie węża:	Collaboration:

• stwarzanie • zwracanie współrzędnych końca węża • wykonywanie tury • reagowanie na otrzymane informacje • zbieranie informacji o sobie • niszczenie węża • ruch do celu • szukanie leku • wywoływanie zwiększania ilości przeżytych tur bakterii • usuwanie swoich informacji ze spota • ustaw czas trwania mocy specjalnej • ustaw czas trwania ogłuszenia • ustaw liczebność bakterii niezbędną do zarażenia z jedzenia • ustaw liczebność bakterii niezbędną do zarażenia nie z jedzenia	• WspI • informacje

Classname: Spot	
Superclass: brak Subclass(es): brak	
Responsibilities: przechowywanie obiektów, które zajmują ten spot oraz hashmap z informacji udostępnianych przez nie: • zwróć informacje o obiektach przechowywane na spocie • zwróć obiekt, który ostatnio pojawił się na spocie i nadal na nim jest • zwróć listę obiektów zajmujących spot • dodaj obiekt do listy obiektów na spocie • dodaj obiekt, który udostępnia dostęp do swoich współrzędnych • usuń obiekt, który udostępnia dostęp do swoich współrzędnych • zwróć obiekt, który udostępnia dostęp do swoich współrzędnych	Collaboration: • ExistingI • informacje

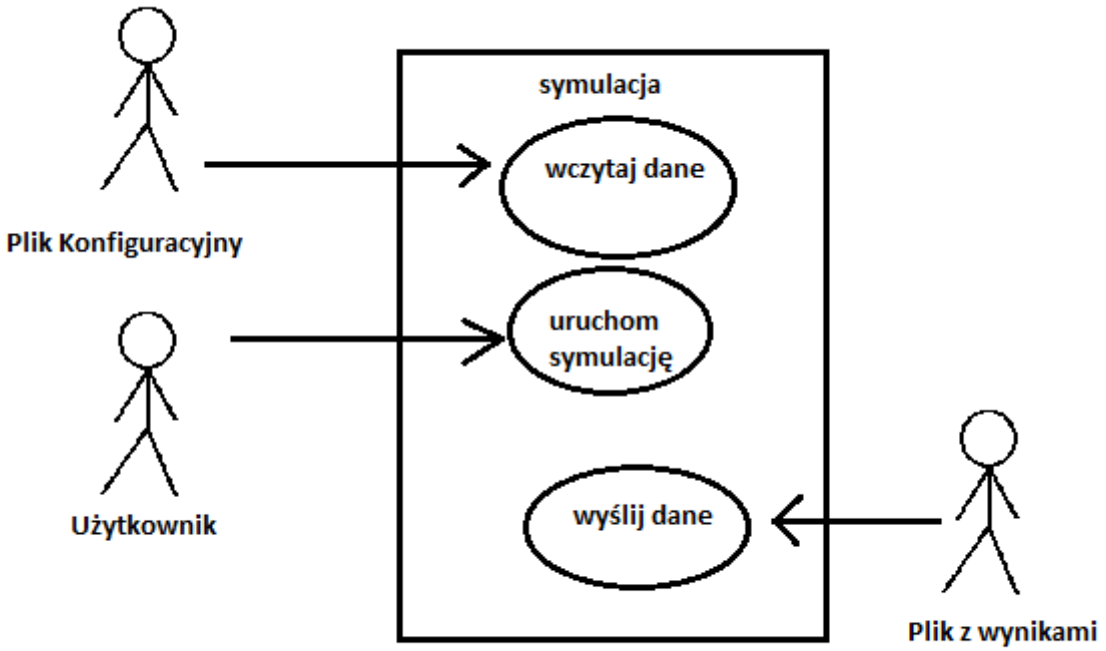
Classname: TylkoPole	
Superclass: brak Subclass(es): brak	
Responsibilities: przechowywanie i zwracanie współrzędnych niezajętych spotów	Collaboration: brak

Classname: WriteData	
Superclass: brak Subclass(es): brak	
Responsibilities: zapisywanie wyników	Collaboration: brak

symulacji do pliku CSV	

Classname: Zyje	
Superclass: Obiekt Subclass(es): Grupa, Waz	
Responsibilities: przeprowadzanie operacji wspólnych dla żywych obiektów: • ruch na sąsiedni spot • interakcja z innymi obiektami zajmującymi spot • ustaw czas trwania zarażenia/rozmnażania się bakterii • ustaw minimalną ilość wygranych, by grupa przeżyła spotkanie z głową węża i zaczęła się rozmnażać na nim • ustaw minimalną ilość wygranych, by grupa na jedzeniu przeżyła spotkanie z głową węża i zaczęła się rozmnażać na nim	Collaboration: brak

Diagram przypadków użycia:



Diagramy aktywności:

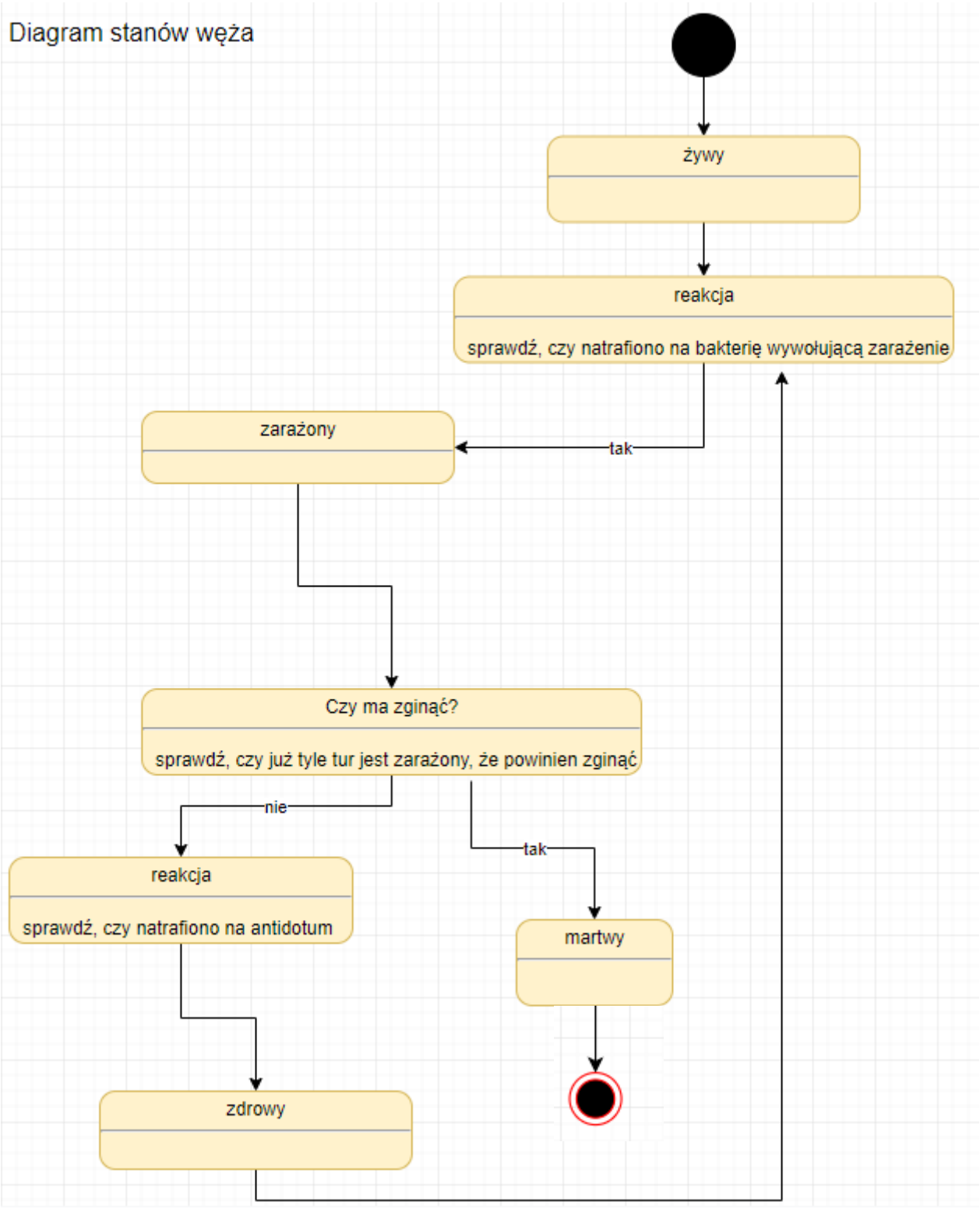
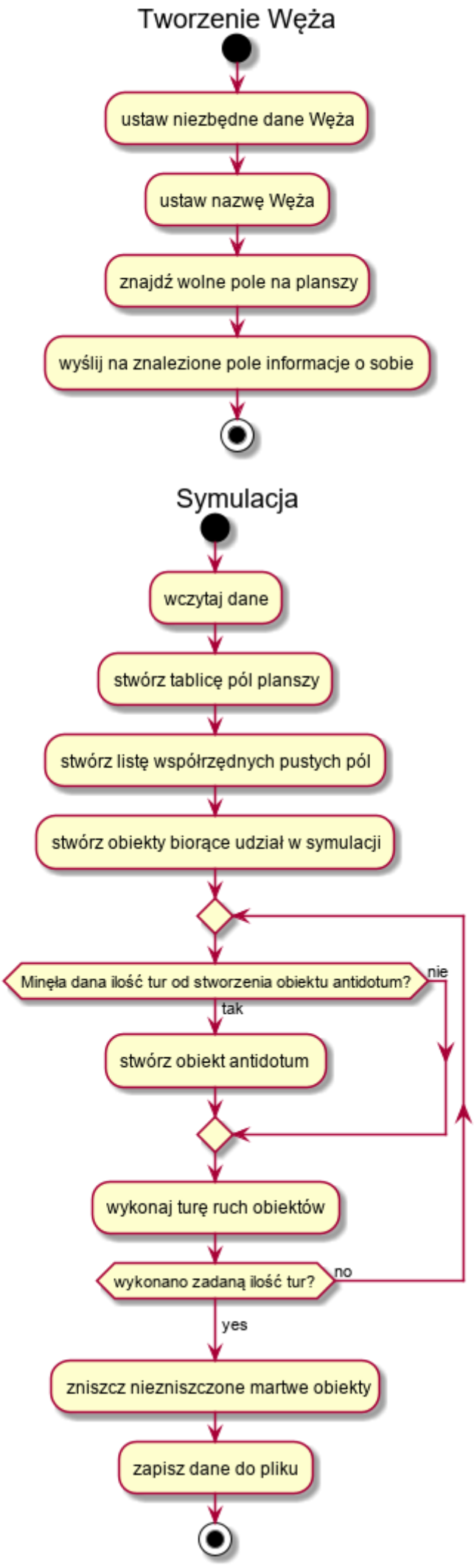


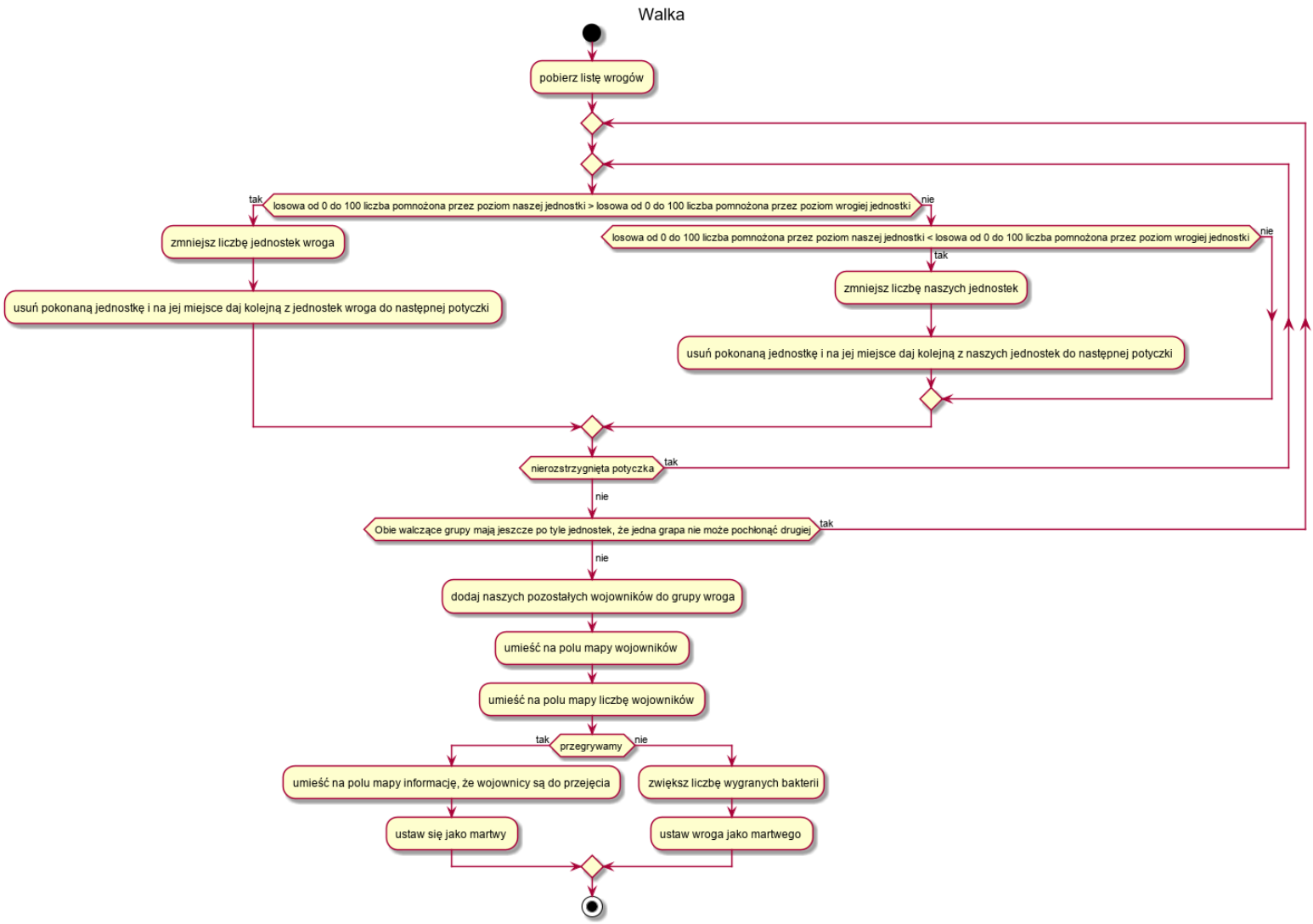
Diagram stanów węża
żywy
reakcja
sprawdź, czy natrafiono na bakterię wywołującą zarażenie
zarażony
tak
Czy ma zginąć?
sprawdź, czy już tyle tur jest zarażony, że powinien zginąć
nie
tak
reakcja
sprawdź, czy natrafiono na antidotum
martwy
zdrowy

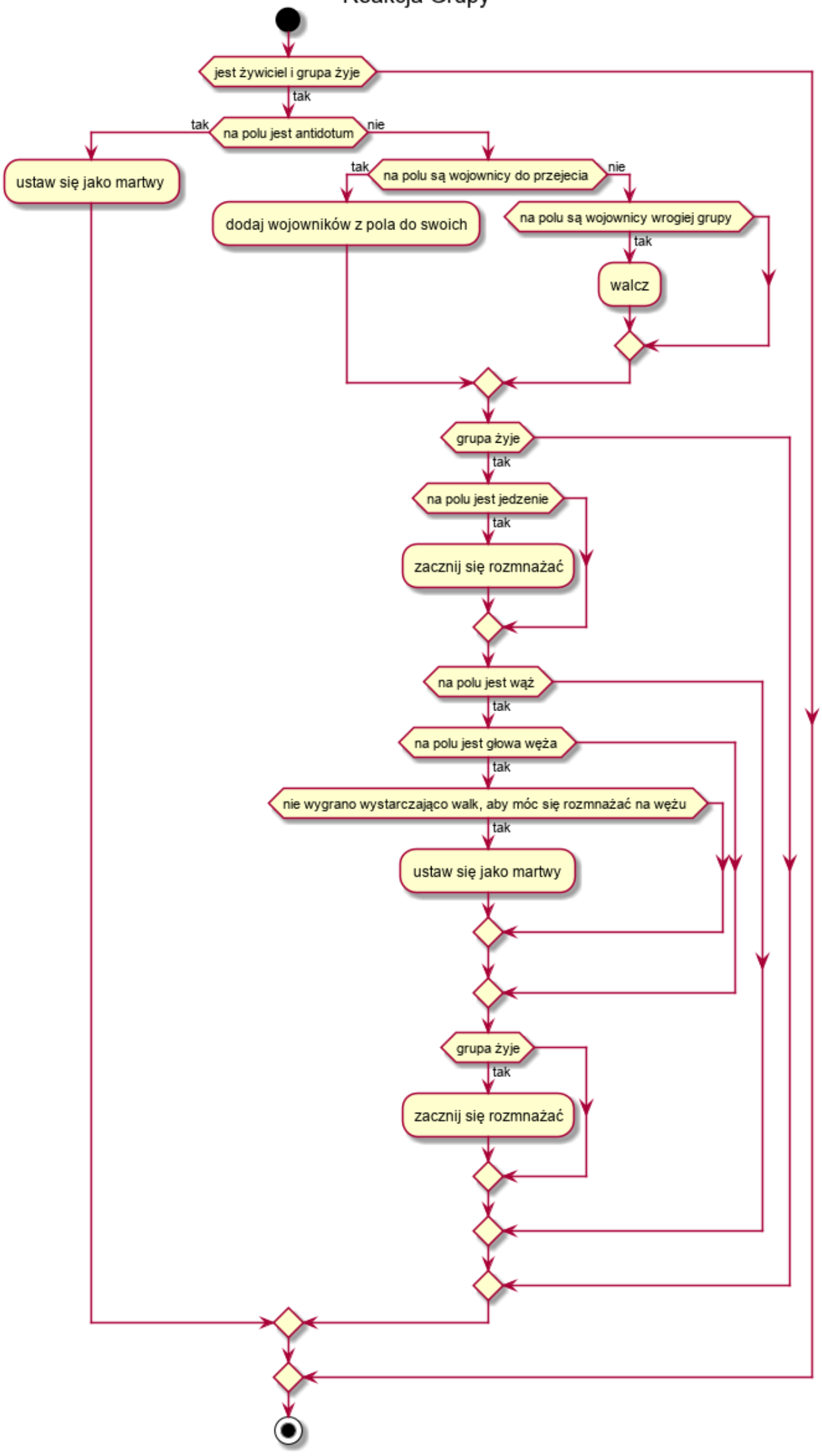

Diagram stanów grupy bakterii
zywy
Czy skończone rozmnażanie?
sprawdź, czy natrafiono na inną grupę
tak
nie
nierozmnażanie
reakcja
sprawdź, czy natrafiono na inną grupę
tak
nie
walka
sprawdź, czy wygrano walkę
nie
tak
martwy
reakcja
sprawdź czy natrafiono na jedzenie
tak
jedzenie
rozmnażanie
nie
reakcja
sprawdź, czy natrafiono na węża
nie
reakcja
sprawdź, czy natrafiono na głowę węża
tak
nie
reakcja
sprawdź, czy można przeżyć to spotkanie z wężem i rozmnażać się na nim
rozmnażanie
tak

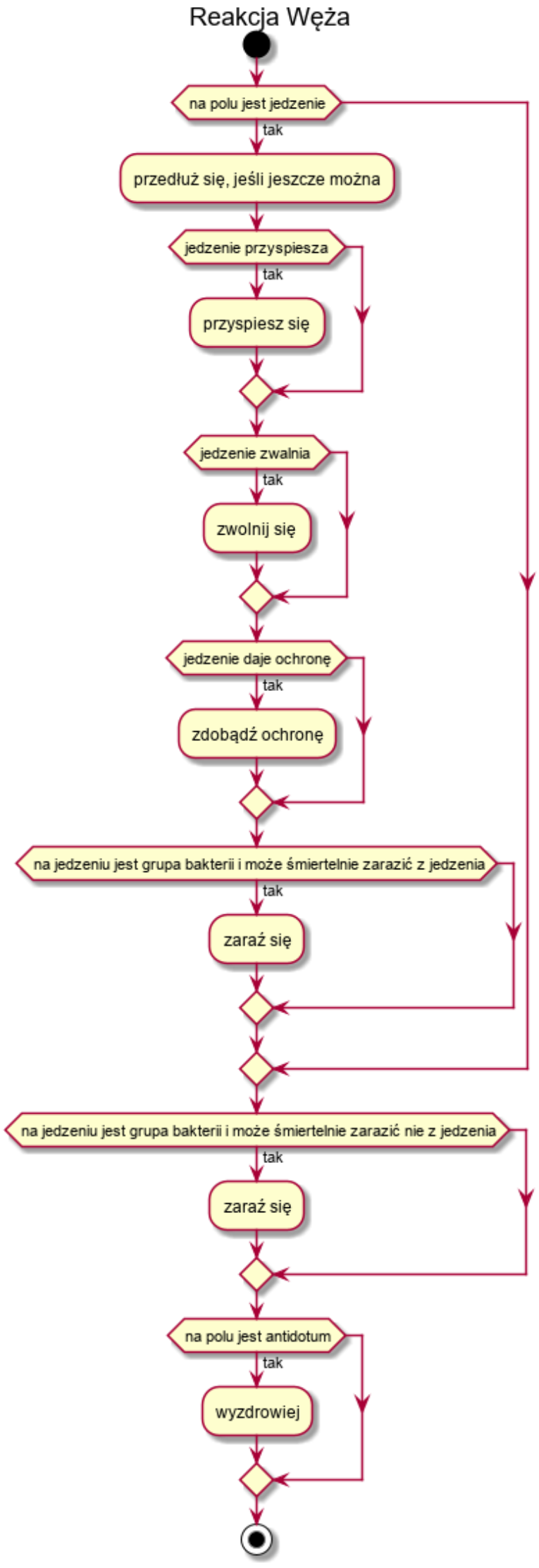
Diagramy Aktywności:

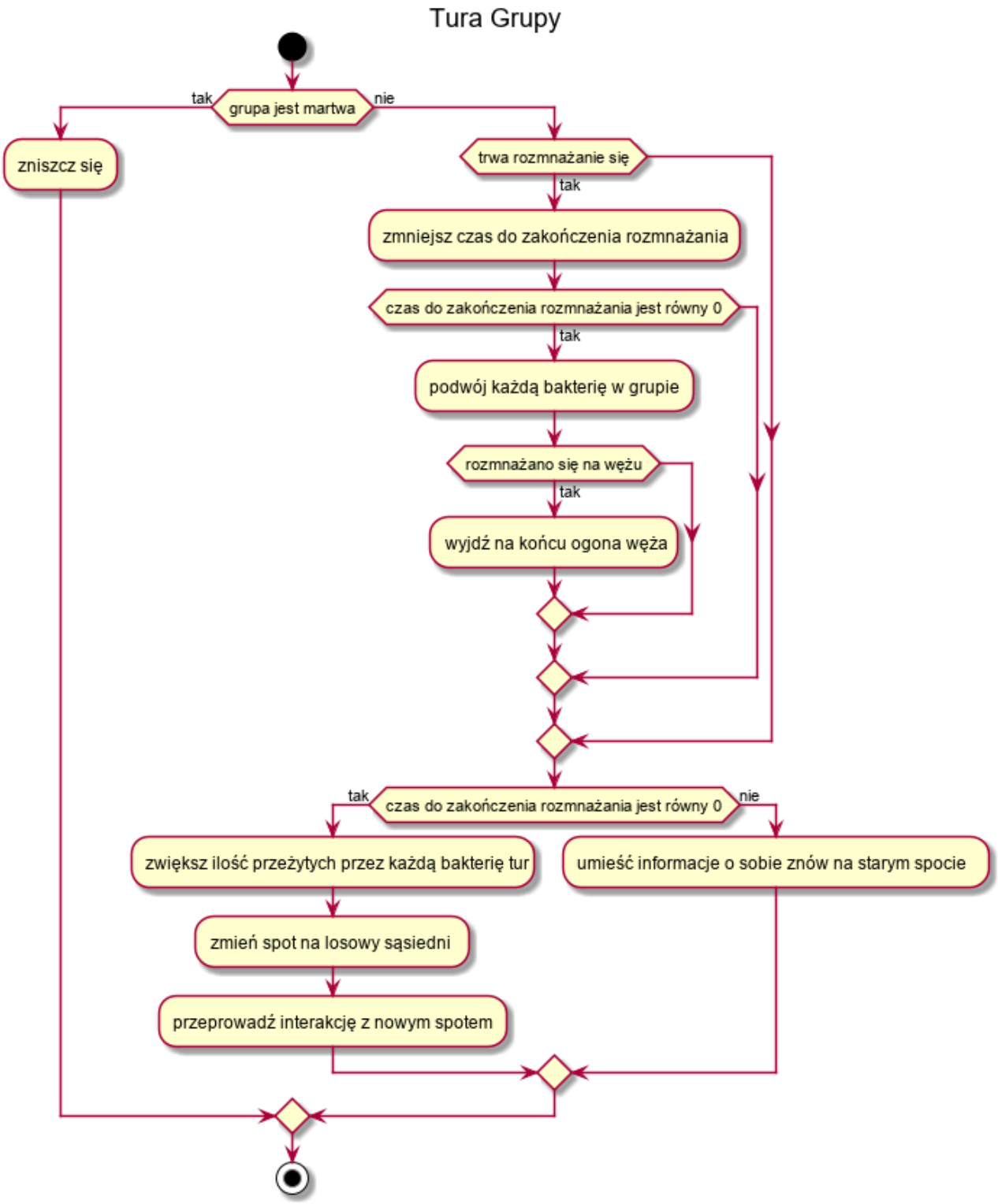

Tworzenie Pakietu Jedzenia
wylosowano możliwość stworzenia specjalnej cechy
tak
dodaj cechę "przyspieszenie"
dodaj cechę "specjalność"
wylosowano możliwość stworzenia specjalnej cechy
tak
dodaj cechę "zwolnienie"
dodaj cechę "specjalność"
Pakiet jedzenia przyspiesza i zwalnia
tak
usuń cechę "przyspieszenie"
usuń cechę "zwolnienie"
usuń cechę "specjalność"
wylosowano możliwość stworzenia specjalnej cechy
tak
dodaj cechę "ochrona"
dodaj cechę "specjalność"
wylosuj wartość, o jaką jedzenie przedłuży węża
ustaw nazwę Pakietu Jedzenia
znajdź wolne pole na planszy
wyslij na znalezione pole informacje o sobie

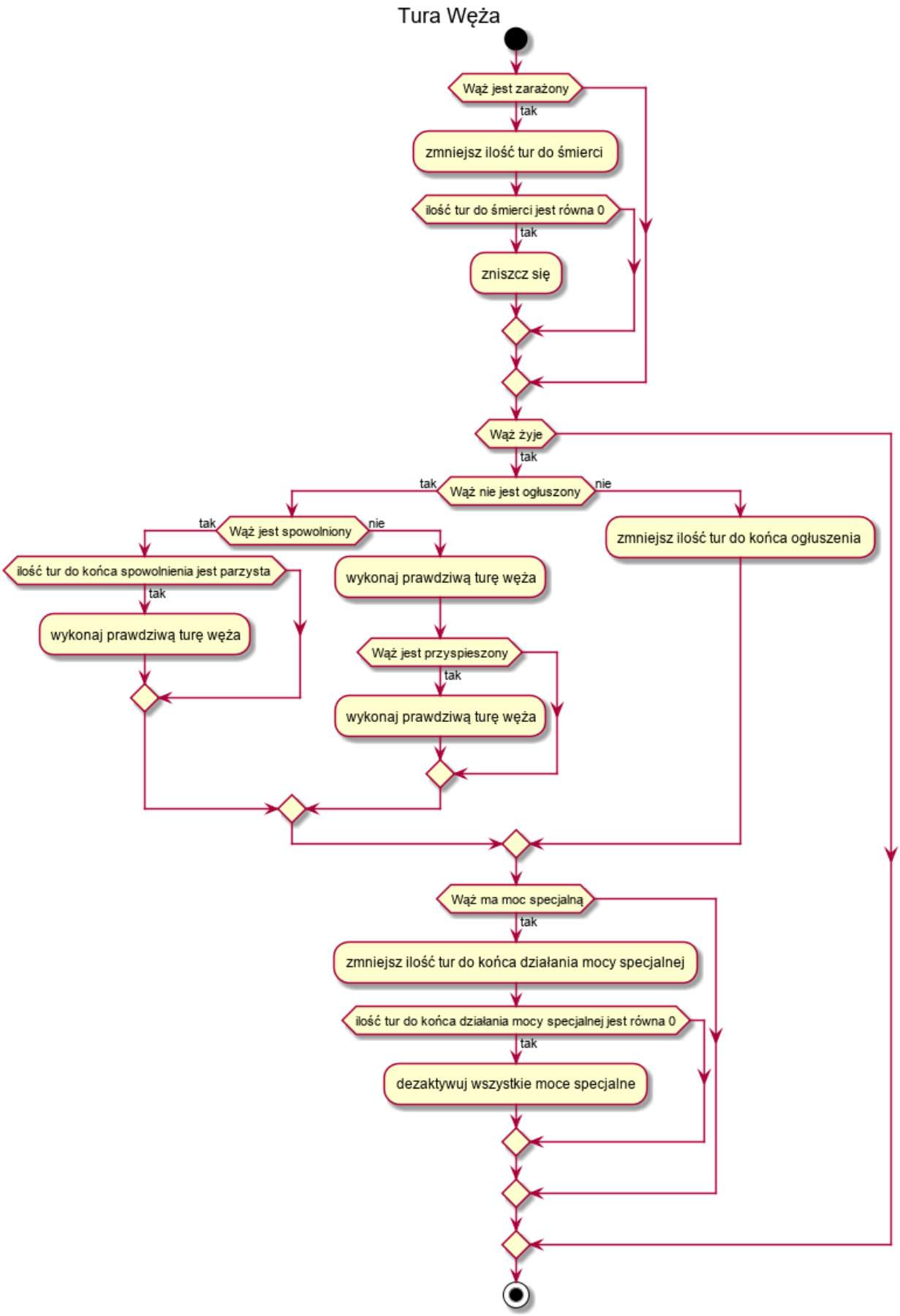

Tworzenie Węża
ustaw niezbędne dane Węża
ustaw nazwę Węża
znajdź wolne pole na planszy
wyślij na znalezione pole informacje o sobie
Symulacja
wczytaj dane
stwórz tablicę pól planszy
stwórz listę współrzędnych pustych pól
stwórz obiekty biorące udział w symulacji
Minęła dana ilość tur od stworzenia obiektu antidotum?
nie
tak
stwórz obiekt antidotum
wykonaj turę ruch obiektów
wykonano zadaną ilość tur?
no
yes
zniszcz niezniszczone martwe obiekty
zapisz dane do pliku

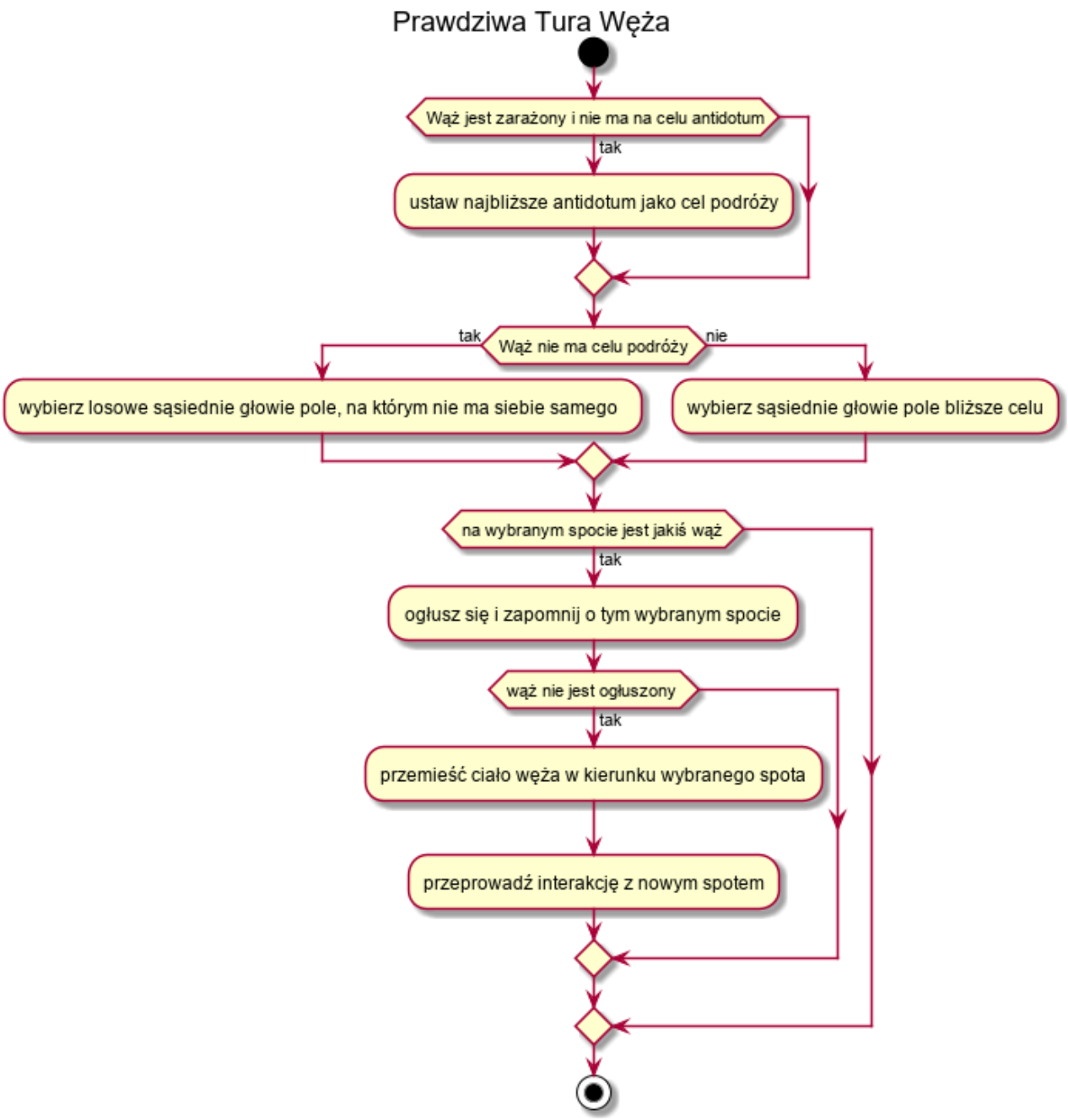

Walka
pobierz listę wrogów
losowa od 0 do 100 liczba pomnożona przez poziom naszej jednostki > losowa od 0 do 100 liczba pomnożona przez poziom wrogiej jednostki
tak
nie
zmniejsz liczbę jednostek wroga
usuń pokonaną jednostkę i na jej miejsce daj kolejną z jednostek wroga do następnej potyczki
losowa od 0 do 100 liczba pomnożona przez poziom naszej jednostki < losowa od 0 do 100 liczba pomnożona przez poziom wrogiej jednostki
tak
nie
zmniejsz liczbę naszych jednostek
usuń pokonaną jednostkę i na jej miejsce daj kolejną z naszych jednostek do następnej potyczki
nierozstrzygnięta potyczka
tak
nie
Obie walczące grupy mają jeszcze po tyle jednostek, że jedna grapa nie może pochłonąć drugiej
tak
nie
dodaj naszych pozostałych wojowników do grupy wroga
umieść na polu mapy wojowników
umieść na polu mapy liczbę wojowników
przegrywamy
tak
nie
umieść na polu mapy informację, że wojownicy są do przejęcia
ustaw się jako martwy
zwiększ liczbę wygranych bakterii
ustaw wroga jako martwego


Reakcja Grupy
jest żywiciel i grupa żyje
tak
na polu jest antidotum
tak
nie
ustaw się jako martwy
na polu są wojownicy do przejecia
tak
nie
dodaj wojowników z pola do swoich
na polu są wojownicy wrogiej grupy
tak
walcz
grupa żyje
tak
na polu jest jedzenie
tak
zacznij się rozmnażać
na polu jest wąż
tak
na polu jest głowa węża
tak
nie wygrano wystarczająco walk, aby móc się rozmnażać na wężu
tak
ustaw się jako martwy
grupa żyje
tak
zacznij się rozmnażać


Reakcja Węża
na polu jest jedzenie
tak
przedłuż się, jeśli jeszcze można
jedzenie przyspiesza
tak
przyspiesz się
jedzenie zwalnia
tak
zwolnij się
jedzenie daje ochronę
tak
zdobądź ochronę
na jedzeniu jest grupa bakterii i może śmiertelnie zarazić z jedzenia
tak
zaraź się
na jedzeniu jest grupa bakterii i może śmiertelnie zarazić nie z jedzenia
tak
zaraź się
na polu jest antidotum
tak
wyzdrowiej


Tura Grupy
grupa jest martwa
tak
nie
zniszcz się
trwa rozmnażanie się
tak
zmniejsz czas do zakończenia rozmnażania
czas do zakończenia rozmnażania jest równy 0
tak
podwój każdą bakterię w grupie
rozmnażano się na wężu
tak
wyjdź na końcu ogona węża
czas do zakończenia rozmnażania jest równy 0
tak
nie
zwiększ ilość przeżytych przez każdą bakterię tur
umieść informacje o sobie znów na starym spocie
zmień spot na losowy sąsiedni
przeprowadź interakcję z nowym spotem


Tura Węża
Wąż jest zarażony
tak
zmniejsz ilość tur do śmierci
ilość tur do śmierci jest równa 0
tak
zniszcz się
Wąż żyje
tak
tak
Wąż nie jest ogłuszony
nie
tak
Wąż jest spowolniony
nie
zmniejsz ilość tur do końca ogłuszenia
ilość tur do końca spowolnienia jest parzysta
tak
wykonaj prawdziwą turę węża
wykonaj prawdziwą turę węża
Wąż jest przyspieszony
tak
wykonaj prawdziwą turę węża
Wąż ma moc specjalną
tak
zmniejsz ilość tur do końca działania mocy specjalnej
ilość tur do końca działania mocy specjalnej jest równa 0
tak
dezaktywuj wszystkie moce specjalne


Prawdziwa Tura Węża
Wąż jest zarażony i nie ma na celu antidotum
tak
ustaw najbliższe antidotum jako cel podróży
tak
Wąż nie ma celu podróży
nie
wybierz losowe sąsiednie głowie pole, na którym nie ma siebie samego
wybierz sąsiednie głowie pole bliższe celu
na wybranym spocie jest jakiś wąż
tak
ogłusz się i zapomnij o tym wybranym spocie
wąż nie jest ogłuszony
tak
przemieść ciało węża w kierunku wybranego spota
przeprowadź interakcję z nowym spotem

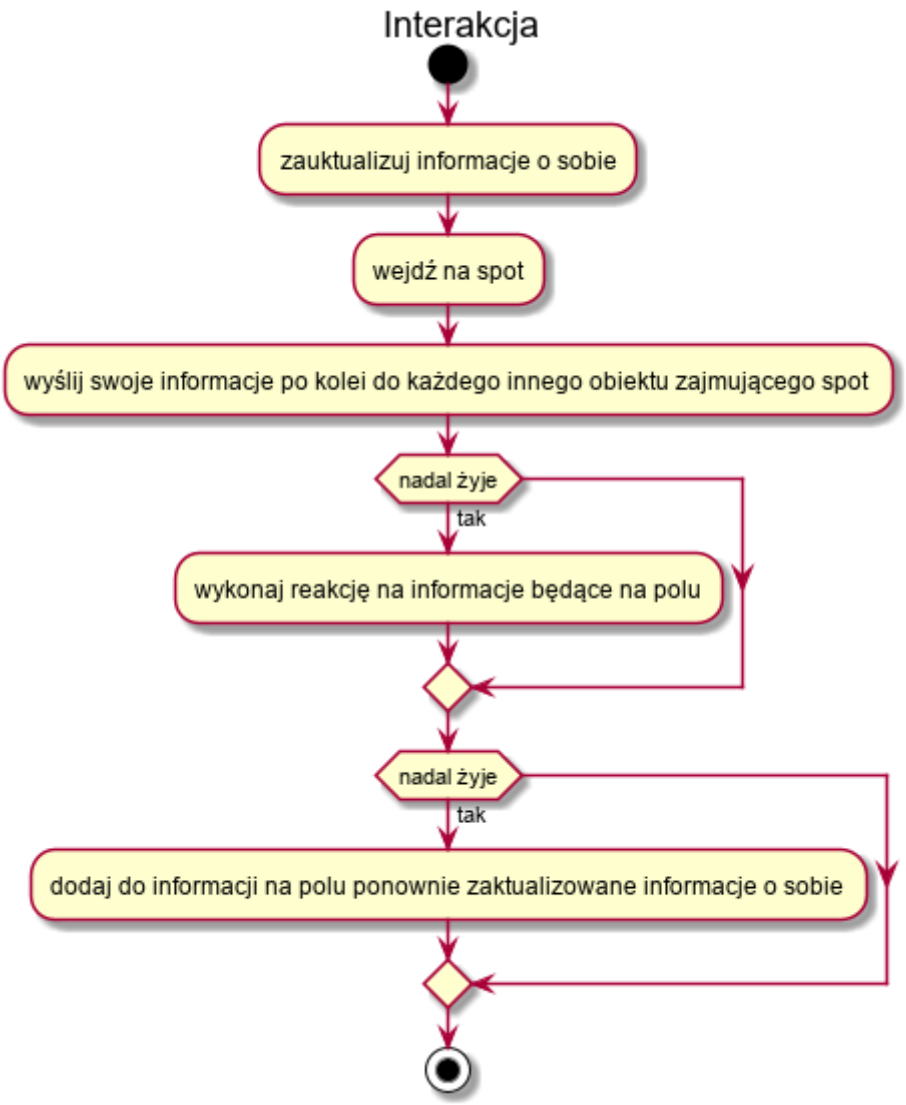

Interakcja
zauktualizuj informacje o sobie
wejdź na spot
wyślij swoje informacje po kolei do każdego innego obiektu zajmującego spot
nadal żyje
tak
wykonaj reakcję na informacje będące na polu
nadal żyje
tak
dodaj do informacji na polu ponownie zaktualizowane informacje o sobie

Diagram sekwencji symulacji:

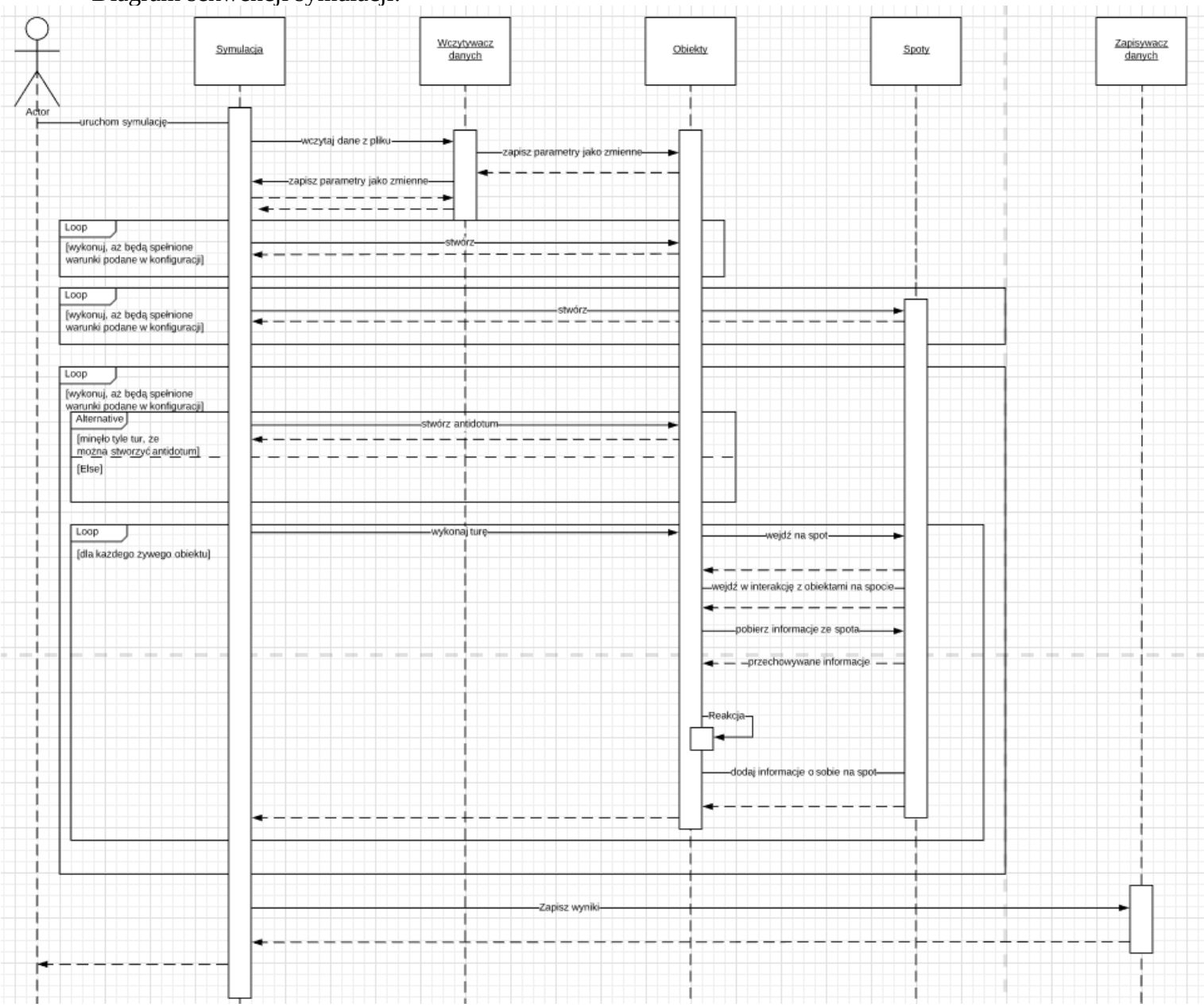
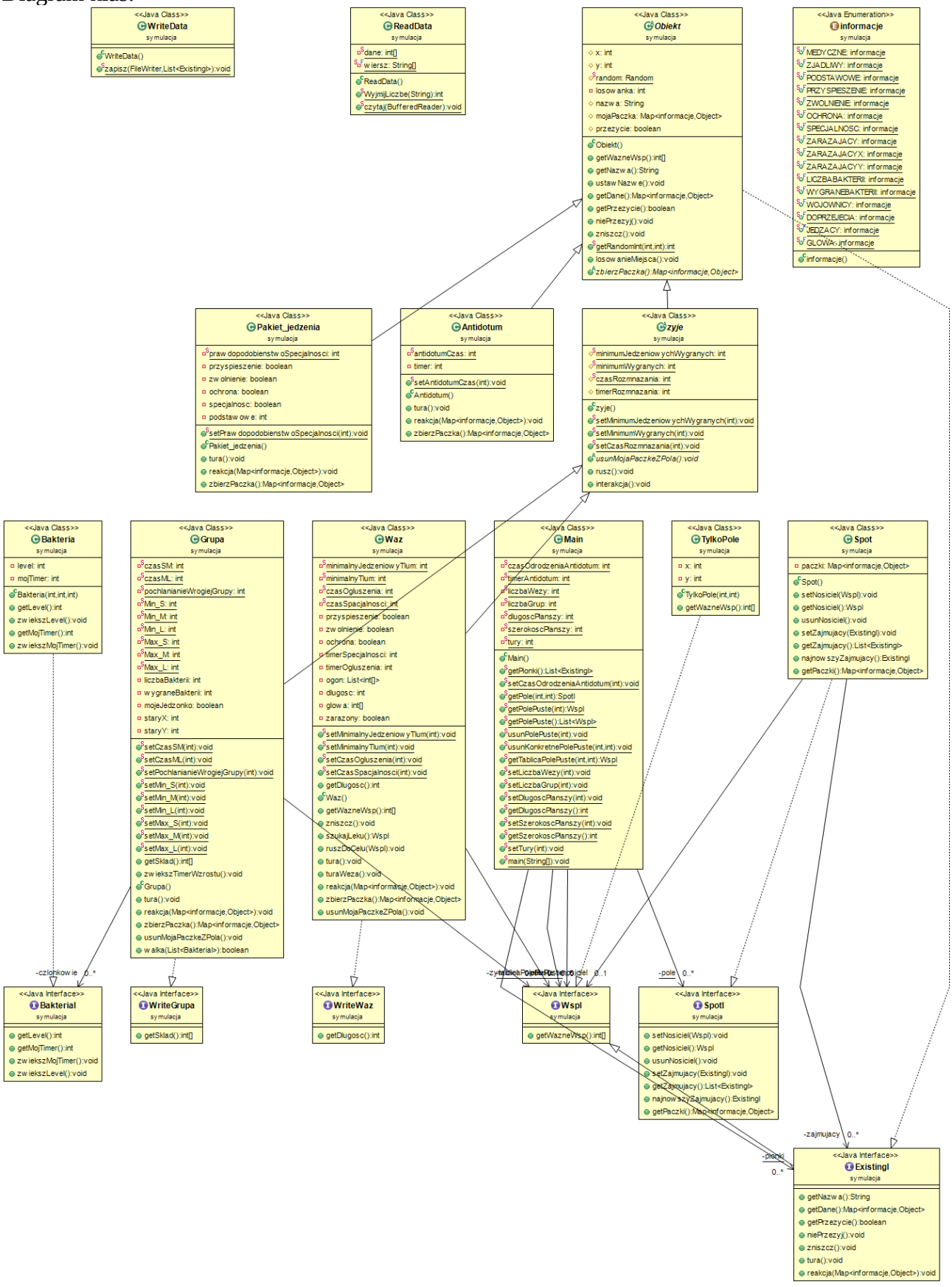


Diagram klas:



Javadoc:

Package symulacja

Interface Summary

Bakterial

Interfejs przechowuj¹cy deklaracje metod u¿ywanych przez bakterie.

ExistingI

Interfejs przechowuj¹cy deklaracje metod u¿ywanych przez Obiekty bior¹ce udzia³ w symulacji.

SpotI

Interfejs daj¹cy dostêp do danych klasy reprezentuj¹cej spoty mapy.

WriteGrupa

Interfejs daj¹cy dostêp klasie zapisuj¹cej wyniki do iloœci bakterii o poszczególnych levelach w grupie.

WriteWaz

Interfejs daj¹cy dostêp klasie zapisuj¹cej wyniki do d³ugoœci wê¿a.

WspI

Interfejs funkcyjny dla wszystkiego, co posiada wspó³rzêdne.

Class Summary

Antidotum

Klasa reprezentuj¹ca antidotum.

Bakteria

Klasa reprezentuj¹ca pojedyncz¹ bakteriê.

Grupa

Klasa reprezentuj¹ca grupê bakterii.

Main

Klasa odpowiadaj¹ca za przeprowadzenie symulacji.

Obiekt

Klasa reprezentuj¹ca cechy wspólne dla ka¿dego Obiektu bior¹cego udzia³ w symulacji.

Pakiet_jedzenia

Klasa reprezentuj¹ca jedzenie.

ReadData

Klasa, której zadaniem jest odczytanie danych z pliku konfiguracyjnego symulacji.

Spot

Klasa reprezentuj¹ca pole mapy, na którym mog¹ byæ Obiekty w symulacji.

TylkoPole

Klasa przechowuj¹ca wspó³rzêdne pustego pola mapy symulacji.

Waz

Klasa reprezentuj¹ca wê¿a.

WriteData

Klasa odpowiadaj¹ca za zapis wyników symulacji do pliku zewnêtrznego.

informacje

Typ wyliczeniowy przechowuj¹cy mo¿liwe do wykorzystania klucze map bêd¹cych zbiorami informacji o Obiektach.

zyje

Klasa reprezentuj¹ca cechy wspólne dla ka¿dego obiektu, który siê porusza.

---

symulacja

Class Antidotum

```
java.lang.Object
    |
    +--Obiekt
        |
        +--symulacja.Antidotum
```

All Implemented Interfaces:

ExistingI

< Constructors > < Methods >

---

public class **Antidotum**
extends Obiekt

Klasa reprezentuj¹ca antidotum.

Author:

Mati

Version:

1.0.0

Constructors

Antidotum

```
public  Antidotum()
```

Stwarza nowe antidotum na mapie symulacji.

Methods

reakcja

`public void` **`reakcja`**`(java.util.Map przyslanaPaczka)`

Okreœla reakcje antidotum na otrzymane informacje.

Parameters:

przyslanaPaczka - - zbiór otrzymanych informacji, na podstawie których bêdzie przeprowadzona reakcja

---

setAntidotumCzas

`public static void` **`setAntidotumCzas`**`(int dane)`

---

tura

`public void` **`tura`**`()`

Przeprowadza turê antidotum.

---

zbierzPaczka

`public java.util.Map` **`zbierzPaczka`**`()`

Zbiera informacje o antidotum, które maj[1] byæ przechowywane na polu mapy, gdzie siê znajduje antidotum.

Overrides:

zbierzPaczka in class Obiekt

---

symulacja

Class Bakteria

```
java.lang.Object
    |
    +--symulacja.Bakteria
```

All Implemented Interfaces:
Bakterial

< Constructors > < Methods >

---

public class **Bakteria**
extends java.lang.Object
implements Bakterial

Klasa reprezentuj¹ca pojedyncz¹ bakteriê.

Author:

Mati

Version:

1.0.0

Constructors

Bakteria

```
public  Bakteria(int i,
                 int srednie,
                 int duze)
```

Stwarza now¹ bakteriê.

Parameters:

i - - okreœla wielkoœæ stworzonej bakterii
srednie - - okreœla licznik ju¿ prze¿ytych tur dla nowej œredniej bakterii
duze - - okreœla licznik ju¿ prze¿ytych tur dla nowej du¿ej bakterii

Methods

getLevel

```
public int getLevel()
```

Zwraca liczbowo wyra¿on¹ wielkoœæ bakterii.

Returns:

liczbowo wyra¿ona wielkoœæ bakterii

---

getMojTimer

```
public int getMojTimer()
```

Zwraca liczbê tur prze¿ytych tur bakterii.

Returns:

liczbowo wyra¿ona wielkoœæ bakterii

---

zwiekszLevel

```
public void zwiekszLevel()
```

Zwiêksza wielkoœæ bakterii.

zwiekszMojTimer

`public void` **`zwiekszMojTimer`**`()`

Zwiêksza liczbê prze¿ytych tur bakterii.

symulacja

Interface Bakterial

< Methods >

public interface **Bakterial**

Interfejs przechowuj¹cy deklaracje metod u¿ywanych przez bakterie.

Author:
Mati

Version:
1.0.0

Methods

getLevel

`public int` **`getLevel`**`()`

Zwraca liczbowo wyra¿on¹ wielkoœæ bakterii.

getMojTimer

`public int` **`getMojTimer`**`()`

Zwraca liczbê tur prze¿ytych tur bakterii.

zwiekszLevel

`public void` **`zwiekszLevel`**`()`

Zwiêksza wielkoœæ bakterii.

zwiekszMojTimer

`public void` **`zwiekszMojTimer`**`()`

Zwiêksza liczbê prze¿ytych tur bakterii.

---

symulacja

Interface ExistingI

All Implemented Interfaces:

WspI

< Methods >

---

public interface **ExistingI**
implements WspI

Interfejs przechowuj¹cy deklaracje metod u¿ywanych przez Obiekty bior¹ce udzia³ w symulacji.

Author:

Mati

Version:

1.0.0

Methods

getDane

`public java.util.Map` **`getDane`**`()`

Zwraca paczkê informacji o Obiekcie.

Returns:

paczka informacji o Obiekcie

---

getNazwa

`public java.lang.String` **`getNazwa`**`()`

Zwraca "nazwê" Obiektu.

Returns:

"nazwa" Obiektu

---

getPrzezycie

`public boolean` **`getPrzezycie`**`()`

Zwraca zmienn¹ logiczn¹ okreœlaj¹c¹, czy Obiekt ¿yje.

Returns:

true - gdy Obiekt ¿yje; false - gdy Obiekt nie ¿yje

---

niePrzezyj

`public void` **`niePrzezyj`**`()`

Zmienn¹ logiczn¹ okreœlaj¹c¹, czy Obiekt ¿yje, ustawia na fa³sz.

---

reakcja

`public void` **`reakcja`**`(java.util.Map przyslanaPaczka)`

Okreœla reakcje Obiektu na otrzymane informacje.

Parameters:

przyslanaPaczka - - zbiór otrzymanych informacji, na podstawie których bêdzie przeprowadzona reakcja

---

tura

`public void` **`tura`**`()`

Przeprowadza turê Obiektu.

---

zniszcz

`public void` **`zniszcz`**`()`

Usuwa Obiekt z listy Obiektów bior¹cych udzia³ w symulacji i z obecnie zajmowanego przez ten Obiekt pola.

symulacja

Class Grupa

```
java.lang.Object
    |
    +--Obiekt
        |
        +--zyje
            |
            +--symulacja.Grupa
```

All Implemented Interfaces:

ExistingI, WriteGrupa

< Constructors > < Methods >

---

public class **Grupa**
extends zyje
implements WriteGrupa

Klasa reprezentuj¹ca grupê bakterii.

Author:

Mati

Version:

1.0.0

Constructors

Grupa

`public` **`Grupa`**`()`

Stwarza now¹ grupê bakterii na mapie symulacji.

Methods

getSklad

`public int[]` **`getSklad`**`()`

Zwraca iloœci bakterii o kolejnych rozmiarach.

Returns:

tablica przechowuj¹ca iloœci bakterii o kolejnych rozmiarach

---

reakcja

`public void` **`reakcja`**`(java.util.Map przyslanaPaczka)`

Okreœla reakcje grupy bakterii na otrzymane informacje.

Parameters:

przyslanaPaczka - - zbiór otrzymanych informacji, na podstawie których bêdzie przeprowadzona reakcja

---

setCzasML

`public static void` **`setCzasML`**`(int dane)`

Ustawia iloœæ tur potrzebn¹ na uroœniêcie bakterii na du¿¹.

Parameters:

dane - - iloœæ tur potrzebna na uroœniêcie bakterii na du¿¹

---

setCzasSM

`public static void` **`setCzasSM`**`(int dane)`

Ustawia iloœæ tur potrzebn¹ na uroœniêcie bakterii na œredni¹.

Parameters:

dane - - iloœæ tur potrzebna na uroœniêcie bakterii a œredni¹

---

setMax_L

`public static void` **`setMax_L`**`(int dane)`

Ustawia maksymaln¹ iloœæ du¿ych bakterii, jaka mo¿e zostaæ stworzona w nowej grupie.

Parameters:

dane - - maksymalna iloœæ du¿ych bakterii, jaka mo¿e zostaæ stworzona w nowej grupie

---

setMax_M

`public static void` **`setMax_M`**`(int dane)`

Ustawia maksymaln¹ iloœæ œrenich bakterii, jaka mo¿e zostaæ stworzona w nowej grupie.

Parameters:

dane - - maksymalna iloœæ œrednich bakterii, jaka mo¿e zostaæ stworzona w nowej grupie

setMax_S

`public static void` **`setMax_S`**`(int dane)`

Ustawia maksymaln¹ iloœæ ma³ych bakterii, jaka mo¿e zostaæ stworzona w nowej grupie.

Parameters:

dane - - maksymalna iloœæ ma³ych bakterii, jaka mo¿e zostaæ stworzona w nowej grupie

setMin_L

`public static void` **`setMin_L`**`(int dane)`

Ustawia minimaln¹ iloœæ du¿ych bakterii, jaka mo¿e zostaæ stworzona w nowej grupie.

Parameters:

dane - - minimalna iloœæ du¿ych bakterii, jaka mo¿e zostaæ stworzona w nowej grupie

setMin_M

`public static void` **`setMin_M`**`(int dane)`

Ustawia minimaln¹ iloœæ œrednich bakterii, jaka mo¿e zostaæ stworzona w nowej grupie.

Parameters:

dane - - minimalna iloœæ œrednich bakterii, jaka mo¿e zostaæ stworzona w nowej grupie

setMin_S

`public static void` **`setMin_S`**`(int dane)`

Ustawia minimaln¹ iloœæ ma³ych bakterii, jaka mo¿e zostaæ stworzona w nowej grupie.

Parameters:

dane - - minimalna iloœæ ma³ych bakterii, jaka mo¿e zostaæ stworzona w nowej grupie

setPochlanianieWrogiejGrupy

`public static void` **`setPochlanianieWrogiejGrupy`**`(int dane)`

Ustawia procent liczby cz³onków grupy bakterii, poni¿ej którego zostaje ona wch³oniêta do grupy, z któr¹ walczy.

Parameters:

dane - - procent liczby cz³onków grupy bakterii, poni¿ej którego zostaje ona wch³oniêta do grupy, z któr¹ walczy

tura

`public void` **`tura`**`()`

Przeprowadza turê grupy bakterii.

usunMojaPaczkeZPola

`public void` **`usunMojaPaczkeZPola`**`()`

Usuwa informacje o grupie bakterii z pola mapy.

Overrides:

usunMojaPaczkeZPola in class zyje

walka

`public boolean` **`walka`**`(java.util.List wrog)`

Przeprowadza walkê miêdzy jedn¹ list¹ przechowuj¹c¹ bakterie a drug¹.

Parameters:

wrog - - lista przecowujaca wrogie jednostki, z ktorymi zostanie stoczona walka

Returns:

true - gdy wygra³a ta grupa bakteii; false - gdy przegra³a ta grupa bakterii

zbierzPaczka

`public java.util.Map` **`zbierzPaczka`**`()`

Zbiera informacje o grupie bakterii.

Overrides:

zbierzPaczka in class Obiekt

zwiekszTimerWzrostu

`public void` **`zwiekszTimerWzrostu`**`()`

Zwiêksza iloœæ przezytych tur ka¿dej bakterii w grupie.

symulacja

Class Main

```
java.lang.Object
    |
    +--symulacja.Main
```

< Constructors > < Methods >

---

public class **Main**
extends java.lang.Object

Klasa odpowiadaj¹ca za przeprowadzenie symulacji.

Author:
Mati

Version:
1.0.0

Constructors

Main

```
public  Main()
```

Methods

getDlugoscPlanszy

```
public static int getDlugoscPlanszy()
```

Zwraca d³ugoœæ mapy symulacji.

Returns:

d³ugoœæ mapy symulacji

---

getPionki

```
public static java.util.List getPionki()
```

Zwraca listê wszystkich Obiektów bior¹cyh udzia³ w symulacji.

Returns:

lista wszystkich Obiektów bior¹cyh udzia³ w symulacji

---

getPole

```
public static SpotI getPole(int x,
                            int y)
```

Zwraca pole mapy symulacji o zadanych wspó³rzêdnych.

Parameters:

x - - wspó³rzêdna x wybranego miejsca mapy symulacji
y - - wspó³rzêdna y wybranego miejsca mapy symulacji

Returns:

konkretne pole mapy symulacji

---

getPolePuste

```
public static java.util.List getPolePuste()
```

Zwraca listê przechowuj¹c¹ puste pola mapy.

Returns:

lista pustych pól

---

getPolePuste

```
public static WspI getPolePuste(int x)
```

Zwraca puste pole z listy przechowuj¹cej puste pola mapy.

Parameters:

x - - liczba okreœlaj¹ca, które puste pole z listy nale¿y zwróciæ

Returns:

puste pole

---

getSzerokoscPlanszy

```
public static int getSzerokoscPlanszy()
```

Zwraca szerokoœæ mapy symulacji.

Returns:

szerokoœæ mapy symulacji

---

getTablicaPolePuste

```
public static WspI getTablicaPolePuste(int x,
                                       int y)
```

Zwraca puste pole o konkretnych wspó³rzêdnych z tablicy przechowuj¹cej puste pola.

Parameters:

x - - wspó³rzêdna x wybranego pustego pola
y - - wspó³rzêdna y wybranego pustego pola

Returns:

puste pole o konkretnych wspó³rzêdnych

---

main

```
public static void main(java.lang.String[] args)
```

Przeprowadza symulacjê.

Parameters:

args - - parametry, które mo¿na przekazaæ w licœcie poleceñ przy uruchomieniu (niewykorzystywane)

---

setCzasOdrodzeniaAntidotum

```
public static void setCzasOdrodzeniaAntidotum(int dane)
```

Ustawia iloœæ tur potrzebn¹ na stworzenie nowego antidotum.

Parameters:

dane - - iloœæ tur potrzebna na stworzenie nowego antidotum.

---

setDlugoscPlanszy

```
public static void setDlugoscPlanszy(int dane)
```

Ustawia d³ugoœæ mapy symulacji.

Parameters:

dane - - d³ugoœæ mapy symulacji

---

setLiczbaGrup

```
public static void setLiczbaGrup(int dane)
```

Ustawia liczbê grup bakterii do stworzenia.

Parameters:

dane - - liczba grup bakterii do stworzenia

---

setLiczbaWezy

```
public static void setLiczbaWezy(int dane)
```

Ustawia liczbê wê¿y do stworzenia.

Parameters:

dane - - liczba wê¿y do stworzenia

---

setSzerokoscPlanszy

```
public static void setSzerokoscPlanszy(int dane)
```

Ustawia szerokoœæ mapy symulacji.

Parameters:

dane - - szerokoœæ mapy symulacji

---

setTury

```
public static void setTury(int dane)
```

Ustawia iloœæ tur symulacji.

Parameters:

dane - - iloœæ tur symulacji

---

usunKonkretnePolePuste

```
public static void usunKonkretnePolePuste(int x,
                                          int y)
```

Usuwa puste pole o konkretnych wspó³rzêdnych z listy przechowuj¹cej puste pola mapy.

Parameters:

x - - wspó³rzêdna x wybranego pustego pola
y - - wspó³rzêdna y wybranego pustego pola

usunPolePuste

```
public static void usunPolePuste(int numerPola)
```

Usuwa puste pole z listy przechowuj¹cej puste pola mapy.

Parameters:

numerPola - - liczba okreœlaj¹ca, które puste pole z listy nale¿y usun¹æ

---

symulacja

Class Obiekt

```
java.lang.Object
    |
    +--symulacja.Obiekt
```

All Implemented Interfaces:
ExistingI

Direct Known Subclasses:
Antidotum, Pakiet_jedzenia, zyje

< Constructors > < Methods >

---

public abstract class **Obiekt**
extends java.lang.Object
implements ExistingI

Klasa reprezentuj¹ca cechy wspólne dla ka¿dego Obiektu bior¹cego udzia³ w symulacji.

Author:
Mati

Version:
1.0.0

Constructors

Obiekt

```
public  Obiekt()
```

Methods

getDane

```
public java.util.Map getDane()
```

Zwraca paczkê informacji o Obiekcie.

Returns:

paczka informacji o Obiekcie

---

getNazwa

```
public java.lang.String getNazwa()
```

Zwraca "nazwê" Obiektu.

Returns:

"nazwa" Obiektu

---

getPrzezycie

```
public boolean getPrzezycie()
```

Zwraca zmienn¹ logiczn¹ okreœlaj¹c¹, czy Obiekt ¿yje.

Returns:

true - gdy Obiekt ¿yje; false - gdy Obiekt nie ¿yje

---

getRandomInt

```
public static int getRandomInt(int min,
                               int max)
```

Losuje liczbê ca³kowit¹ z podanego przedzia³u.

Parameters:

min - - najmniejsza liczba mo¿liwa do wylosowania
max - - najwiêksza liczba mo¿liwa do wylosowania

Returns:

zwraca liczbê ca³kowit¹ wylosowan¹ z podanego przedzia³u

---

getWazneWsp

`public int[] getWazneWsp()`

Zwraca wspó³rzêdne x i y Obiektu.

Returns:

tablica przechowuj¹ca wspó³rzêdne x i y Obiektu

---

losowanieMiejsca

`public void losowanieMiejsca()`

Usuwa losowe z pustych pól planszy. Wspó³rzêdne tego pola zapisuje jako wspó³rzêdne danego Obiektu.

---

niePrzezyj

`public void niePrzezyj()`

Zmienn¹ logiczn¹ okreœlaj¹c¹, czy Obiekt ¿yje, ustawia na fa³sz.

---

ustawNazwe

`public void ustawNazwe()`

Ustawia nazwê klasy obiektu jako jego "nazwê".

---

zbierzPaczka

`public abstract java.util.Map zbierzPaczka()`

Zbiera informacje o Obiekcie.

Returns:

mapa przechowuj¹ca informacje o Obiekcie

---

zniszcz

`public void zniszcz()`

Usuwa Obiekt z listy Obiektów bior¹cych udzia³ w symulacji i z obecnie zajmowanego przez ten Obiekt pola.

symulacja

Class Pakiet_jedzenia

```
java.lang.Object
    |
    +--Obiekt
        |
        +--symulacja.Pakiet_jedzenia
```

All Implemented Interfaces:

ExistingI

< Constructors > < Methods >

---

public class **Pakiet_jedzenia**
extends Obiekt

Klasa reprezentuj¹ca jedzenie.

Author:

Mati

Version:

1.0.0

Constructors

Pakiet_jedzenia

```
public  Pakiet_jedzenia()
```

Stwarza nowe jedzenie na mapie symulacji.

Methods

reakcja

```
public void reakcja(java.util.Map przyslanaPaczka)
```

Okreœla reakcje jedzenia na otrzymane informacje.

Parameters:

przyslanaPaczka - - zbiór otrzymanych informacji, na podstawie których bêdzie przeprowadzona reakcja

---

setPrawdopodobienstwoSpecjalnosci

`public static void` **`setPrawdopodobienstwoSpecjalnosci`**`(int dane)`

Ustawia prawdopodobieñstwo wyst¹pienia w jedzeniu cechy specjalnej.

Parameters:

dane - - prawdopodobieñstwo wyst¹pienia w jedzeniu cechy specjalnej

---

tura

`public void` **`tura`**`()`

Przeprowadza turê jedzenia.

---

zbierzPaczka

`public java.util.Map` **`zbierzPaczka`**`()`

Zbiera informacje o jedzeniu, które maj¹ byæ przechowywane na polu mapy, gdzie siê znajduje antidotum.

Overrides:

zbierzPaczka in class Obiekt

---

symulacja

Class ReadData

```
java.lang.Object
    |
    +--symulacja.ReadData
```

< Constructors > < Methods >

---

public class **ReadData**
extends java.lang.Object

Klasa, której zadaniem jest odczytanie danych z pliku konfiguracyjnego symulacji.

Author:

Mati

Version:

1.0.0

Constructors

ReadData

```
public  ReadData()
```

Methods

WyjmijLiczbe

```
public static int WyjmijLiczbe(java.lang.String linijka)
                  throws java.lang.NumberFormatException
```

Wyjmuje liczbê bêd¹c¹ na ostatnim miejscu w linijce tekstu.

Parameters:

linijka - - tekst, który zawiera liczbê

Returns:

liczba bêd¹ca na koñcu linijki tekstu

Throws:

java.lang.NumberFormatException - wyrzucany, kiedy na koñcu tekstu nie znajduje siê liczba

---

czytaj

```
public static void czytaj(java.io.BufferedReader in)
                  throws java.io.IOException
```

Ustawia wartoœci pól w plikach klasowych symulacji na podstawie danych z pliku tekstowego.

Parameters:

in - - plik, z którego odczytywane s¹ dane

Throws:

java.io.IOException - wyrzucany, kiedy nie ma dostêpu do pliku, z którego maj¹ byæ odczytane dane

---

symulacja

Class Spot

```
java.lang.Object
    |
    +--symulacja.Spot
```

All Implemented Interfaces:

SpotI

---

public class **Spot**
extends java.lang.Object
implements SpotI

Klasa reprezentuj¹ca pole mapy, na którym mog¹ byæ Obiekty w symulacji.

Author:
Mati

Version:
1.0.0

Constructors

Spot

`public` **`Spot`**`()`

Stwarza nowe pole mapy, na którym mog¹ byæ Obiekty w symulacji.

Methods

getNosiciel

`public` `WspI` **`getNosiciel`**`()`

Zwraca nosiciela zajmuj¹cego pole.

Returns:
dane - nosiciel zajmuj¹cy pole

---

getPaczki

`public java.util.Map` **`getPaczki`**`()`

Zwraca zbiór informacji o obiektach znajduj¹cych siê na polu.

Returns:
zbiór informacji o obiektach znajduj¹cych siê na polu

getZajmujacy

`public java.util.List` **`getZajmujacy`**`()`

Zwraca listê obiektów zajmuj¹cych pole mapy symulacji.

Returns:

lista obiektów zajmuj¹cych pole mapy symulacji

---

najnowszyZajmujacy

`public` [`ExistingI`] **`najnowszyZajmujacy`**`()`

Zwraca ostatnio dodany obiekt z listy obiektów zajmuj¹cych pole mapy symulacji.

Returns:

ostatnio dodany obiekt z listy obiektów zajmuj¹cych pole mapy symulacji

---

setNosiciel

`public void` **`setNosiciel`**`(`[`WspI`] `dane)`

Ustawia nosiciela na polu, aby jakiœ Obiekt móg³ go przyj¹æ i mieæ dostêp do jego wspó³rzêdnych.

Parameters:

dane - - nosiciel zajmuj¹cy pole

---

setZajmujacy

`public void` **`setZajmujacy`**`(`[`ExistingI`] `dane)`

Dodaje obiekt do listy obiektów znajduj¹cych siê na polu mapy symulacji.

Parameters:

dane - - obiekt, który wchodzi na pole mapy symulacji.

---

usunNosiciel

`public void` **`usunNosiciel`**`()`

Usuwa nosiciela zajmuj¹cego pole.

symulacja

Interface SpotI

< Methods >

public interface **SpotI**

Interfejs daj¹cy dostêp do danych klasy reprezentuj¹cej spoty mapy.

Author:
Mati

Version:
1.0.0

Methods

getNosiciel

`public WspI getNosiciel()`

Zwraca nosiciela zajmuj¹cego pole.

Returns:

dane - nosiciel zajmuj¹cy pole

---

getPaczki

`public java.util.Map getPaczki()`

Zwraca zbiór informacji o obiektach znajduj¹cych siê na polu.

Returns:

zbiór informacji o obiektach znajduj¹cych siê na polu

---

getZajmujacy

`public java.util.List getZajmujacy()`

Zwraca listê obiektów zajmuj¹cych pole mapy symulacji.

Returns:

lista obiektów zajmuj¹cych pole mapy symulacji

---

najnowszyZajmujacy

`public ExistingI najnowszyZajmujacy()`

Zwraca ostatnio dodany obiekt z listy obiektów zajmuj¹cych pole mapy symulacji.

Returns:

ostatnio dodany obiekt z listy obiektów zajmuj¹cych pole mapy symulacji

---

setNosiciel

`public void setNosiciel(WspI dane)`

Ustawia nosiciela na polu, aby jakiœ Obiekt móg³ go przyj¹æ i mieæ dostêp do jego wspó³rzêdnych.

Parameters:

dane - - nosiciel zajmuj¹cy pole

---

setZajmujacy

`public void setZajmujacy(ExistingI dane)`

Dodaje obiekt do listy obiektów znajduj¹cych siê na polu mapy symulacji.

Parameters:

dane - - obiekt, który wchodzi na pole mapy symulacji.

---

usunNosiciel

`public void usunNosiciel()`

Usuwa nosiciela zajmuj¹cego pole.

---

symulacja

Class TylkoPole

```
java.lang.Object
    |
    +--symulacja.TylkoPole
```

All Implemented Interfaces:

WspI

< Constructors > < Methods >

public class **TylkoPole**
extends java.lang.Object
implements WspI

Klasa przechowuj¹ca wspó³rzêdne pustego pola mapy symulacji.

Author:
Mati

Version:
1.0.0

Constructors

TylkoPole

```
public  TylkoPole(int x,
                  int y)
```

Stwarza nowe puste pole.

Parameters:

x - - wspó³rzêdna x pustego pola
y - - wspó³rzêdna y pustego pola

Methods

getWazneWsp

```
public int[] getWazneWsp()
```

Zwraca wspó³rzêdne x i y pustego pola.

Returns:

tablica przechowuj¹ca wspó³rzêdne x i y pustego pola

---

symulacja

Class Waz

```
java.lang.Object
    |
    +--Obiekt
        |
        +--zyje
            |
            +--symulacja.Waz
```

All Implemented Interfaces:
ExistingI, WriteWaz

< Constructors > < Methods >

public class **Waz**
extends zyje
implements WriteWaz

Klasa reprezentuj¹ca wê¿a.

Author:
Mati

Version:
1.0.0

Constructors

Waz

`public` **`Waz`**`()`

Stwarza nowego wê¿a na mapie symulacji.

Methods

getDlugosc

`public int` **`getDlugosc`**`()`

Zwraca d³ugoœæ wê¿a.

Returns:

d³ugoœæ wê¿a

---

getWazneWsp

`public int[]` **`getWazneWsp`**`()`

Zwraca tablicê zawieraj¹c¹ wspó³rzêdne koñca wê¿a.

Returns:

tablica zawieraj¹c¹ wspó³rzêdne koñca wê¿a

Overrides:

getWazneWsp in class Obiekt

---

reakcja

`public void` **`reakcja`**`(java.util.Map przyslanaPaczka)`

Okreœla reakcje wê¿a na otrzymane informacje.

Parameters:

przyslanaPaczka - - zbiór otrzymanych informacji, na podstawie których bêdzie przeprowadzona reakcja

---

ruszDoCelu

`public void` **`ruszDoCelu`**`(WspI cel)`

Zmienia wspó³rzêdne g³owy wê¿a na s¹siednie, bli¿sze celu.

Parameters:

cel - - cel, do którego zmierza w¹¿

---

setCzasOgluszenia

`public static void` **`setCzasOgluszenia`**`(int dane)`

Ustawia iloœæ tur, jak¹ trwa og³uszenie wê¿a.

Parameters:

dane - - iloœæ tur, jak¹ trwa og³uszenie wê¿a

---

setCzasSpacjalnosci

`public static void` **`setCzasSpacjalnosci`**`(int dane)`

Ustawia iloœæ tur, jak¹ trwa specjalna moc z jedzenia.

Parameters:

dane - - iloœæ tur, jak¹ trwa specjalna moc z jedzenia

---

setMinimalnyJedzeniowyTlum

`public static void` **`setMinimalnyJedzeniowyTlum`**`(int dane)`

Ustawia minimaln¹ liczbê bakterii w grupie bakterii znajduj¹cej siê na jedzeniu, aby mog³y one zaraziæ wê¿a.

Parameters:

dane - - minimalna liczbê bakterii w grupie bakterii znajduj¹cej siê na jedzeniu, aby mog³y one zaraziæ wê¿a

---

setMinimalnyTlum

`public static void` **`setMinimalnyTlum`**`(int dane)`

Ustawia minimaln¹ liczbê bakterii w grupie bakterii, aby mog³y one zaraziæ wê¿a.

Parameters:

dane - - minimalna liczbê bakterii w grupie bakterii, aby mog³y one zaraziæ wê¿a

---

szukajLeku

`public` WspI **`szukajLeku`**`()`

Szuka najbli¿czego obiektu bêd¹cego antidotum.

Returns:

najbli¿szy obiekt bêd¹cy antidotum.

---

tura

`public void` **`tura`**`()`

Przeprowadza pocz¹tek tury wê¿a.

---

turaWeza

`public void` **`turaWeza`**`()`

Przeprowadza w³aœciwe rozwiniêcie tury wê¿a.

---

usunMojaPaczkeZPola

`public void` **`usunMojaPaczkeZPola`**`()`

Usuwa z pola mapy symulacji informacjê o byciu tam g³owy wê¿a.

Overrides:

usunMojaPaczkeZPola in class zyje

---

zbierzPaczka

`public java.util.Map` **`zbierzPaczka`**`()`

Zbiera informacje o wê¿u.

Overrides:

zbierzPaczka in class Obiekt

---

zniszcz

`public void` **`zniszcz`**`()`

Usuwa wê¿a z listy Obiektów bior¹cych udzia³ w symulacji i ze wszystkich zajmowanych przez niego pól razem z informacjami o nim z tych pól.

Overrides:

zniszcz in class Obiekt

---

symulacja

Class WriteData

```
java.lang.Object
    |
    +--symulacja.WriteData
```

< Constructors > < Methods >

---

public class **WriteData**
extends java.lang.Object

Klasa odpowiadaj¹ca za zapis wyników symulacji do pliku zewnêtrznego.

Author:

Mati

Version:

1.0.0

Constructors

WriteData

`public` **`WriteData`**`()`

Methods

zapisz

```
public static void zapisz(java.io.FileWriter in,
                          java.util.List pionki)
                   throws java.io.IOException
```

Zapisuje wyniki symulacji do pliku zewnêtrznego.

Parameters:

in - - plik, do którego maj¹ byæ zapisane wyniki symulacji.
pionki - - obiekty, które pozosta³y w symulacji po jej zakoñczeniu

Throws:

java.io.IOException - wyrzucany, kiedy nie mo¿na uzyskaæ dostêpu do pliku, w którym maj¹ byæ zapisane wyniki

---

symulacja

Interface WriteGrupa

< Methods >

---

public interface **WriteGrupa**

Interfejs daj¹cy dostêp klasie zapisuj¹cej wyniki do iloœci bakterii o poszczególnych levelach w grupie.

Author:
Mati

Version:
1.0.0

Methods

getSklad

```
public int[] getSklad()
```

Zwraca iloœci bakterii o kolejnych rozmiarach.

Returns:

tablica przechowuj¹ca iloœci bakterii o kolejnych rozmiarach

---

symulacja

Interface WriteWaz

< Methods >

public interface **WriteWaz**

Interfejs daj¹cy dostêp klasie zapisuj¹cej wyniki do d³ugoœci wê¿a.

Author:
Mati

Version:
1.0.0

Methods

getDlugosc

```
public int getDlugosc()
```

Zwraca d³ugoœæ wê¿a.

Returns:

d³ugoœæ wê¿a

symulacja

Interface WspI

< Methods >

public interface **WspI**

Interfejs funkcyjny dla wszystkiego, co posiada wspó³rzêdne.

Author:
Mati

Version:
1.0.0

Methods

getWazneWsp

`public int[] getWazneWsp()`

Zwraca tablicê wa¿nych wspó³rzêdnych.

Returns:

tablica wa¿nych wspó³rzêdnych

---

symulacja

Class informacje

```
java.lang.Object
    |
    +--java.lang.Enum
        |
        +--symulacja.informacje
```

All Implemented Interfaces:

java.io.Serializable, java.lang.Comparable

< Fields > < Methods >

---

public final class **informacje**
extends java.lang.Enum

Typ wyliczeniowy przechowuj¹cy mo¿liwe do wykorzystania klucze map bêd¹cych zbiorami informacji o Obiektach.

Author:

Mati

Version:

1.0.0

Fields

DOPRZEJECIA

`public static final informacje DOPRZEJECIA`

GLOWA

`public static final informacje GLOWA`

JEDZACY

public static final informacje **JEDZACY**

LICZBABAKTERII

public static final informacje **LICZBABAKTERII**

MEDYCZNE

public static final informacje **MEDYCZNE**

OCHRONA

public static final informacje **OCHRONA**

PODSTAWOWE

public static final informacje **PODSTAWOWE**

PRZYSPIESZENIE

public static final informacje **PRZYSPIESZENIE**

SPECJALNOSC

public static final informacje **SPECJALNOSC**

WOJOWNICY

public static final informacje **WOJOWNICY**

WYGRANEBAKTERII

public static final informacje **WYGRANEBAKTERII**

ZARAZAJACY

public static final informacje **ZARAZAJACY**

ZARAZAJACYX

public static final informacje **ZARAZAJACYX**

ZARAZAJACYY

public static final informacje **ZARAZAJACYY**

ZJADLIWY

public static final informacje **ZJADLIWY**

ZWOLNIENIE

public static final informacje **ZWOLNIENIE**

Methods

valueOf

public static informacje **valueOf**(java.lang.String name)

---

values

public static symulacja.informacje[] **values**()

---

symulacja

Class zyje

```
java.lang.Object
    |
    +--Obiekt
        |
        +--symulacja.zyje
```

All Implemented Interfaces:
ExistingI

Direct Known Subclasses:
Grupa, Waz

< Constructors > < Methods >

public abstract class **zyje**
extends Obiekt

Klasa reprezentuj¹ca cechy wspólne dla ka¿dego obiektu, który siê porusza.

Author:
Mati

Version:
1.0.0

Constructors

zyje

```
public  zyje()
```

Methods

interakcja

```
public void interakcja()
```

Przeprowadza interakcjê obiektu z obiektami znajduj¹cymi siê na tym samym polu mapy symulacji, czyli wywo³uje w nich reakcjê na informacje od tego obiektu oraz wywo³uje w tym obiekcie reakcjê na informacje ju¿ zawarte na polu mapy symulacji, do których nastêpnie dodaje w³asne informacje.

---

rusz

```
public void rusz()
```

Ustawia nowe wspó³rzêdne obiektu, losowe i zarazem s¹siednie do starych.

---

setCzasRozmnazania

```
public static void setCzasRozmnazania(int dane)
```

Ustawia iloœæ tur potrzebn¹ do rozmno¿enia jednostki zara¿aj¹cej bêd¹c¹ te¿ d³ugoœci¹ trwania zara¿enia.

Parameters:

dane - - iloœæ tur potrzebna do rozmno¿enia jednostki zara¿aj¹cej bêd¹c¹ te¿ d³ugoœci¹ trwania zara¿enia

setMinimumJedzeniowychWygranych

`public static void` **`setMinimumJedzeniowychWygranych`**`(int dane)`

Ustawia minimaln¹ liczbê wygranych walk zara¿aj¹cej jednostki znajduj¹cej siê na jedzeniu potrzebn¹, aby mog³a zaraziæ i przeżyæ.

Parameters:

dane - - minimalna liczba wygranych walk zara¿aj¹cej jednostki znajduj¹cej siê na jedzeniu potrzebna, aby mog³a zaraziæ i przeżyæ

setMinimumWygranych

`public static void` **`setMinimumWygranych`**`(int dane)`

Ustawia minimaln¹ liczbê wygranych walk zara¿aj¹cej jednostki potrzebn¹, aby mog³a zaraziæ i przeżyæ.

Parameters:

dane - - minimalna liczba wygranych walk zara¿aj¹cej jednostki potrzebna, aby mog³a zaraziæ i przeżyæ

usunMojaPaczkeZPola

`public abstract void` **`usunMojaPaczkeZPola`**`()`

Usuwa informacje charakterystyczne dla danej klasy z pola mapy symulacji, które jest zajmowane przez obiekt tej klasy.