Dokumentacja projektu – programowanie obiektowe

Autorzy:

Kamil Gawlik

Sebastian Dorabiała

Spis treści

Opis projektu

Tematem projektu jest symulacja wojny. Po mapie, której każde pole to unikalna prowincja, poruszają się armie przejmując prowincje. Każda prowincja może zawierać cenny zasób taki jak zwiększenie liczby żołnierzy lub obrażeń podczas bitwy armii. Przejęcie prowincji przez armię jest symbolizowane zmianą koloru prowincji z szarego (prowincja niczyja) lub koloru innej armii na kolor odpowiadający armii przejmującej (prowincja przejęta). Armie mogą się spotykać, dochodzi wtedy do bitwy i w jej wyniku każda armia traci część swoich żołnierzy. Armie mogą również zbierać cenne zasoby z prowincji, które właśnie przejęły. Czas trwania symulacji jest kontrolowany za pomocą tur, podczas jednej tury każda z armii wykonuje ruch o jedno pole, ewentualnie przeprowadza bitwę z inną armią lub zbiera zasób, a po zakończeniu każdej tury aktualizowana jest mapa. Dodatkowo program mierzy czas rzeczywisty trwania symulacji. Podczas każdej tury do pliku tekstowego zapisywane są aktualne parametry symulacji takie jak współrzędne każdej armii, jej liczebność a w przypadku wystąpienia zdarzeń takich jak zebranie zasobu lub bitwa – wydarzenia te również są zapisywane.

Użytkownik wprowadza dane początkowe przed każdą symulacją takie jak limit tur, rozmiar mapy, współrzędne początkowe czy liczbę żołnierzy każdej armii. Symulacja może zakończyć się na dwa sposoby:

- zwycięstwo militarne – w przypadku gdy na mapie zostanie tylko jedna armia, wtedy jest ona zwycięzcą symulacji,
- zwycięstwo przez limit tur – w przypadku, gdy osiągnięty zostanie limit tur a na mapie zostaną co najmniej dwie armie – wygrywa wtedy armia, która przejęła najwięcej prowincji.

Projekt został napisany w języku C++ przy użyciu środowiska Visual Studio 2019.

Link do internetowego repozytorium projektu:
https://github.com/Kmgt1337/Projekcik-PO

Analiza czasownikowo-rzeczownikowa

Obiekt opisujący armię. Potrzebna jest możliwości poruszania armią na mapie, oraz zbierania zasobów z mapy. Armię będzie opisywać liczba żołnierzy, oddziały wojskowe, nazwa, symbol, pozycje x i y, aktywność.

Klasa opisująca mapę. Klasa będzie aktualizowana wraz z przebiegiem każdej tury. Klasa będzie odpowiadała za rysowanie mapy tzn. obiektów takich jak armie oraz prowincje.

Obiekt opisujący prowincje. Obiekt będzie posiadał właściwości takie jak: współrzędne x i y na mapie, przynależność, symbol, armie w prowincji. Prowincja będzie posiadać własną nazwę.

Klasa odpowiedzialna za zapis wszystkich pożądanych parametrów do pliku.

Klasa odpowiedzialna za przeprowadzenie symulacji. Klasa będzie inicjalizowała symulację oraz przeprowadzała ją.

Klasa odpowiedzialna za sprawdzanie warunków zakończenia symulacji. Klasa będzie sprawdzała czy nastąpiło jedno z dwóch możliwych zakończeń oraz wyznaczała armię zwycięską.

Klasa odpowiedzialna za przeprowadzanie bitw pomiędzy armiami. Klasa będzie przeprowadzała bitwy pomiędzy tylko dwoma armiami oraz wyznaczała jej straty procentowe.

Klasa opisująca Zegar. Klasa będzie liczyła liczbę tur symulacji oraz wyznaczała czas rzeczywisty przeprowadzenia symulacji.

Obiekt opisujący Zasób. Obiekt będzie losować rodzaj zasobu opisujący go oraz przekazywał go armii.

Klasa abstrakcyjna opisująca Oddział. Klasa będzie przekazywać modyfikatory dla armii podczas bitwy.

Obiekt dziedziczący po klasie abstrakcyjnej Oddzial opisujący Artylerię. Obiekt będzie przekazywać modyfikator do ofensywy podczas bitwy.

Obiekt dziedziczący po klasie abstrakcyjnej Oddzial opisujący Ciężką Jazdę. Obiekt będzie przekazywać modyfikator do obrażeń podczas bitwy.

Obiekt dziedziczący po klasie abstrakcyjnej Oddzial opisujący Generała. Obiekt będzie przekazywać modyfikator do obrażeń podczas bitwy.

Obiekt dziedziczący po klasie abstrakcyjnej Oddzial opisujący Lekarza. Obiekt będzie przekazywać modyfikator do liczebności podczas bitwy.

Obiekt dziedziczący po klasie abstrakcyjnej Oddzial opisujący Lekką Jazdę. Obiekt będzie przekazywać modyfikator do defensywy podczas bitwy.

Obiekt dziedziczący po klasie abstrakcyjnej Oddzial opisujący Zwiad. Obiekt będzie przekazywać modyfikator do liczebności podczas bitwy. Obiekt będzie zdawał raport z zwiadu.

Karty CRC

Armia	
• Poruszanie armii po mapie • Zbieranie zasobów z mapy	• Oddzial • Artyleria • General • LekkaJazda • CiezkaJazda • Lekarz • Zwiad • Mapa • rodzajeZasobu • Zasob • Prowincja

Abstract **Oddzial** Artyleria, General, LekkaJazda, CiezkaJazda, Lekarz, Zwiad	
• Dodawanie armii modyfikatorów podczas bitwy	

Artyleria	Oddzial
• Dodawanie armii modyfikatora do ofensywy podczas bitwy	• Armia

CiezkaJazda	Oddzial
• Dodawanie armii modyfikatora do obrażeń podczas bitwy	• Armia

General	Oddzial
• Dodawanie armii modyfikatora do obrażeń podczas bitwy	• Armia

Lekarz	Oddzial
• Dodawanie armii modyfikatora do liczebności podczas bitwy	• Armia

LekkaJazda	Oddzial
• Dodawanie armii modyfikatora do defensywy podczas bitwy	• Armia

Zwiad	Oddzial
• Dodawanie armii modyfikatora do liczebności podczas bitwy • Przeprowadzanie rozpoznania dla armii przed bitwą	• Armia

Operator_bitwy	
• Przeprowadzanie bitw pomiędzy dwiema armiami • Wyliczanie strat w liczebności armii biorących udział w bitwie • Wyznaczanie wygranego i przegranego danej bitwy • Liczenie ilości przeprowadzonych bitw	• Armia

OperatorPliku	
• Zapisywanie do pliku wszystkich parametrów symulacji	• Armia • rodzajeZasobu • Zegar • OperatorZakonczenia • Mapa • OperatorBitwy • OperatorSymulacji

OperatorSymulacji	
• Wczytanie początkowych parametrów od użytkownika • Sprawdzenie poprawności wprowadzonych danych • Przeprowadzenie symulacji	• Armia • Mapa • Zegar • OperatorZakonczenia • OperatorPliku • rodzajeZasobu • Operator_bitwy

OperatorZakonczenia	
• Kontrolowanie czy nastąpiły warunki zakończenia symulacji • Przechowywanie maksymalnej liczby tur • Wybieranie zwycięzcy symulacji	• Armia • Mapa • Zegar

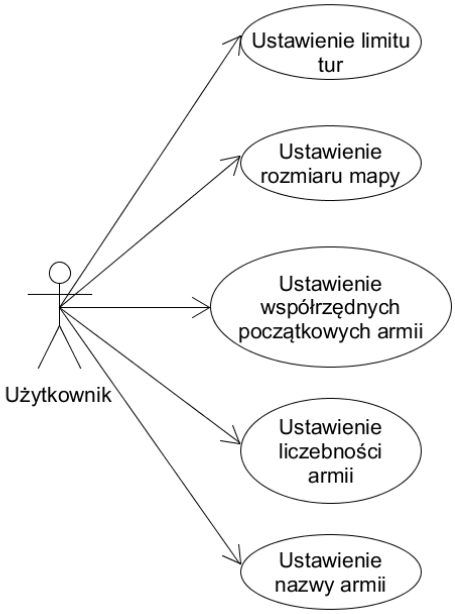
Prowincja	
• Przechowywanie danych o prowincji	• Zasob

Zasob	
• Losowanie rodzaju zasobu • Kontrolowanie aktywności zasobu	• rodzajeZasobu

Mapa	
• Rysowanie na ekranie konsoli mapy (prowincji) i armii • Czyszczenie ekranu po każdym przebiegu symulacji • Aktualizacja mapy po każdym przebiegu symulacji • Przechowywanie informacji o rozmiarze mapy	• Armia • Prowincja

Zegar	
• Kontrola i zmiana liczby tur • Zliczanie rzeczywistego czasu trwania symulacji	

Diagramy przypadków użycia



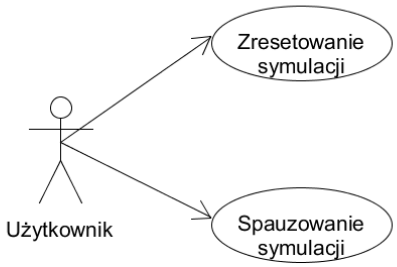


Diagram klas

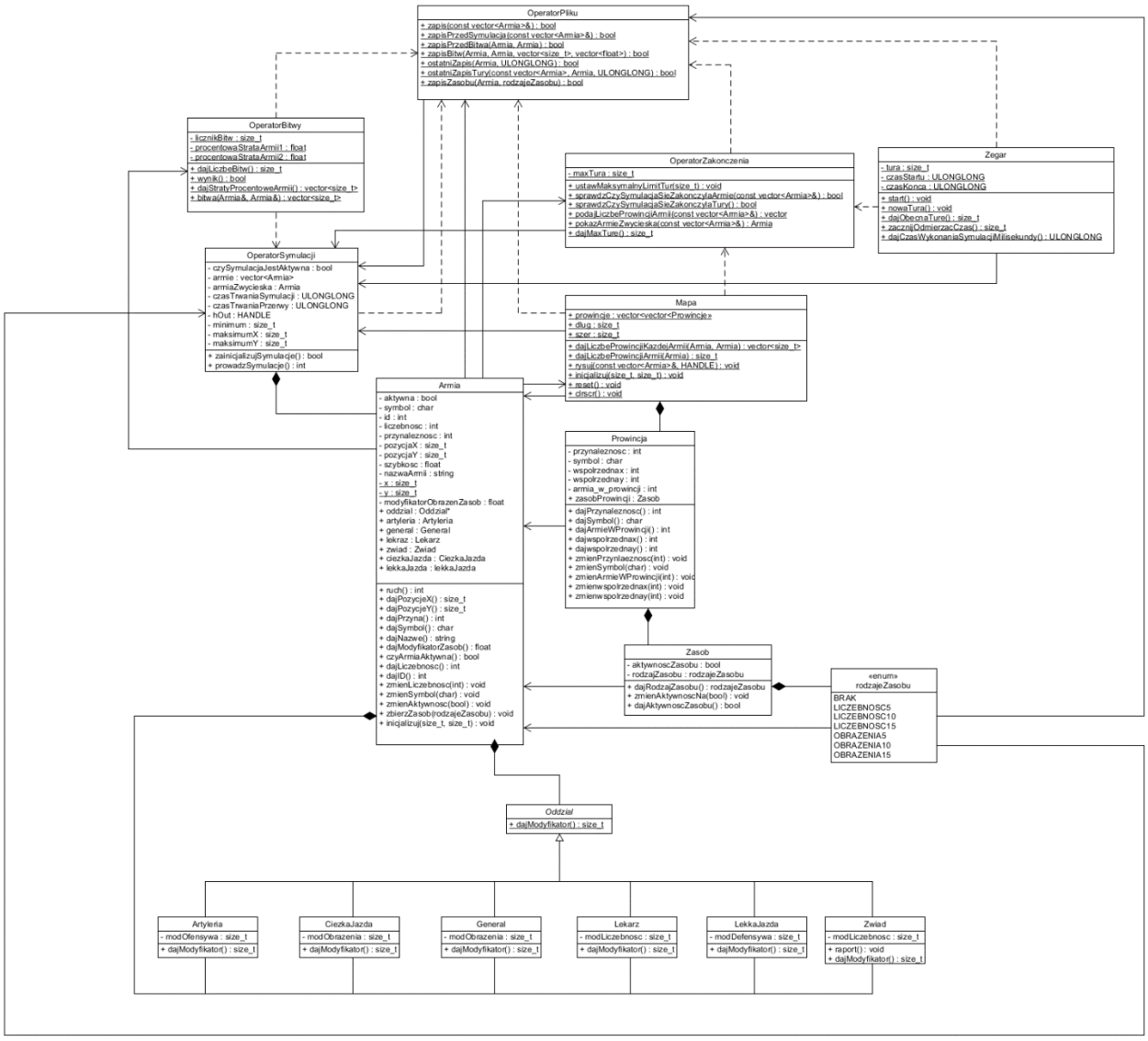
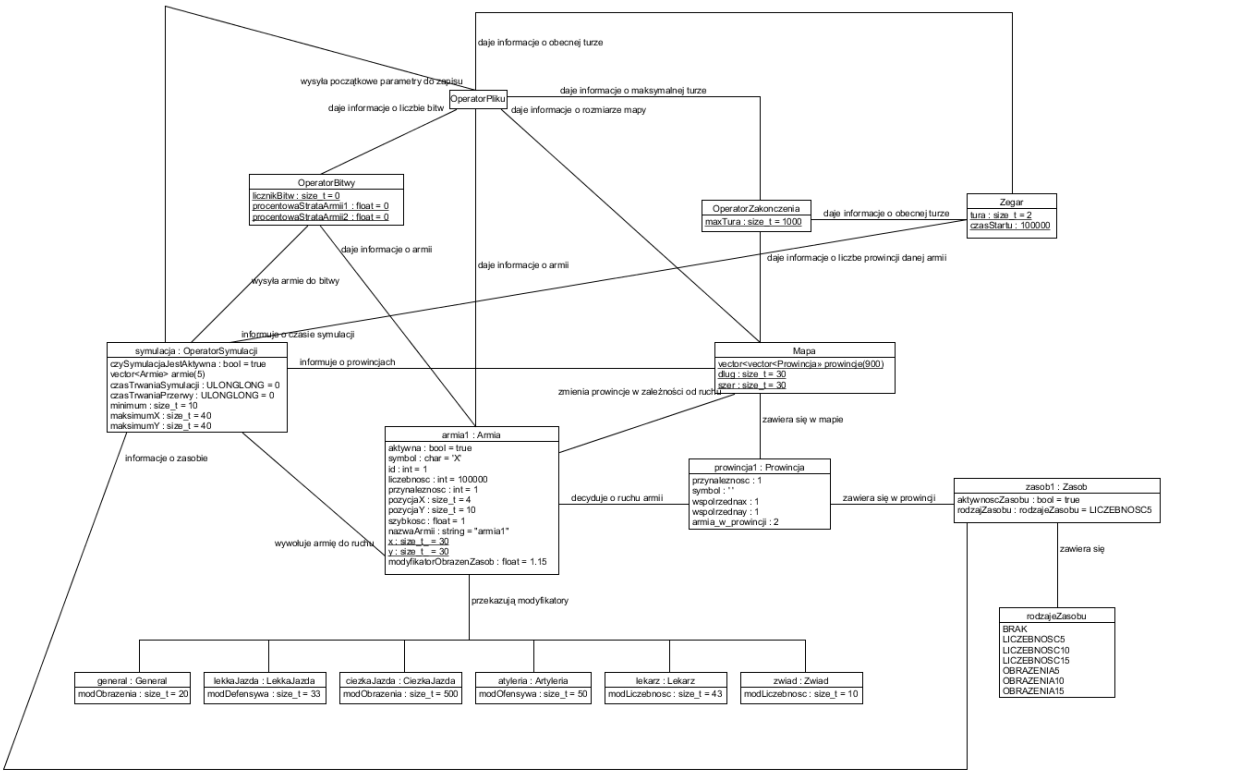
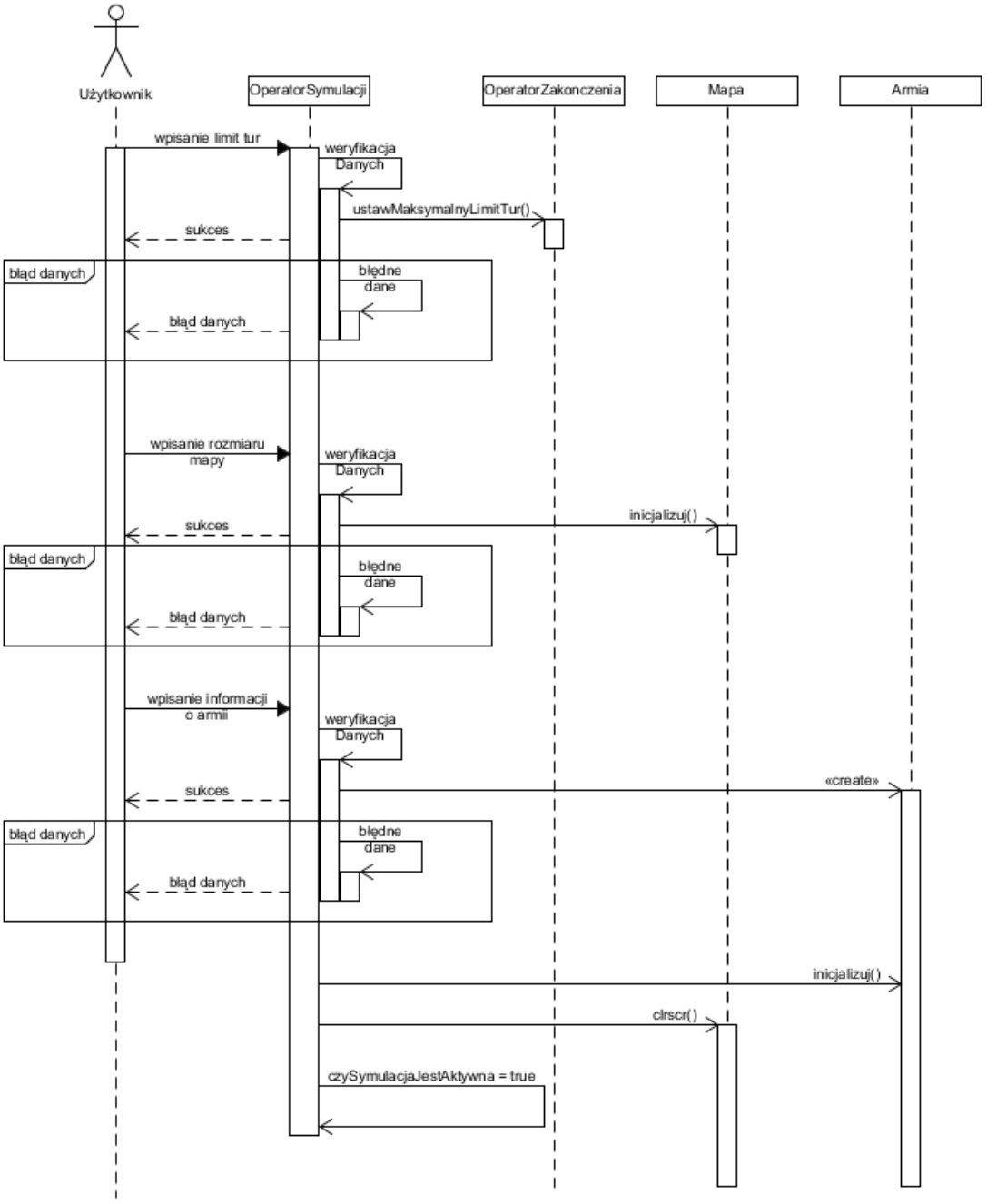
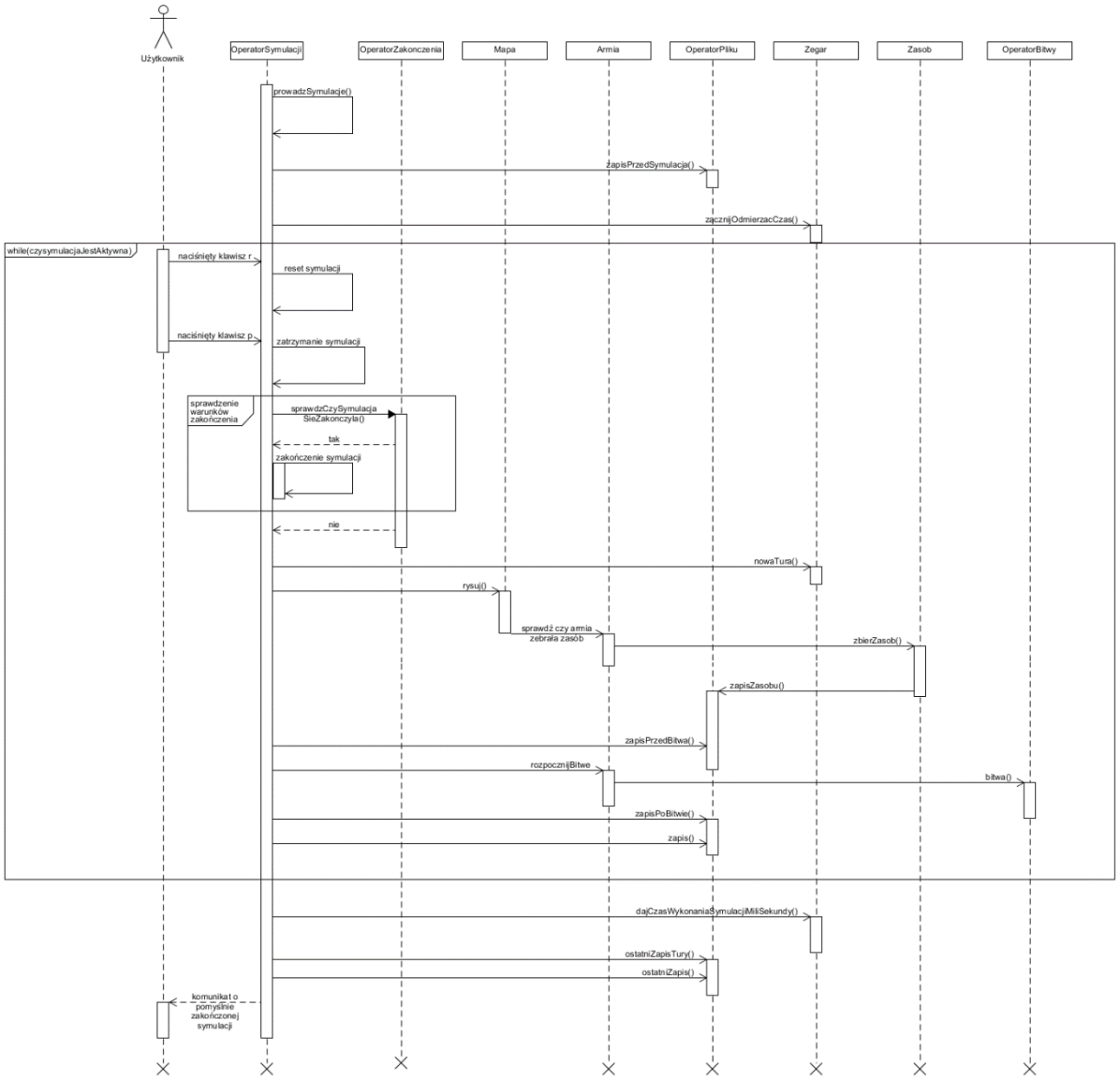


Diagram obiektów

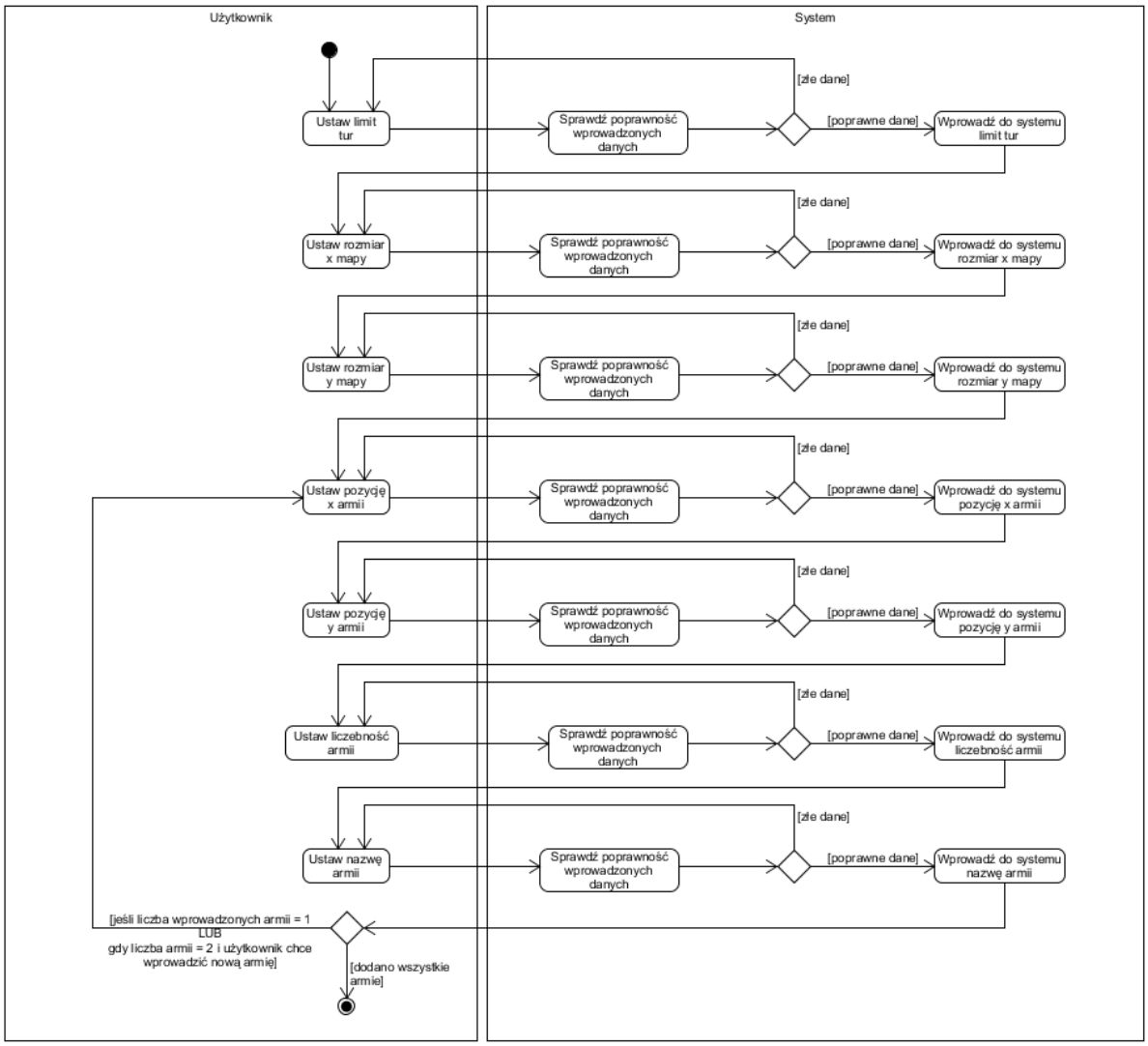


Diagramy sekwencji




Użytkownik
OperatorSymulacji
OperatorZakonczenia
Mapa
Armia
OperatorPliku
Zegar
Zasob
OperatorBitwy
prowadzSymulacje()
zapisPrzedSymulacja()
zacznijOdmierzacCzas()
while(czysymulacjaJestAktywna)
naciśnięty klawisz r
reset symulacji
naciśnięty klawisz p
zatrzymanie symulacji
sprawdzenie warunków zakończenia
sprawdzCzySymulacja SieZakonczyla()
tak
zakończenie symulacji
nie
nowaTura()
rysuj()
sprawdź czy armia zebrała zasób
zbierzZasob()
zapisZasobu()
zapisPrzedBitwa()
rozpocznijBitwe
bitwa()
zapisPoBitwie()
zapis()
dajCzasWykonaniaSymulacjiMiliSekundy()
ostatniZapisTury()
ostatniZapis()
komunikat o pomyślnie zakończonej symulacji

Diagramy aktywności



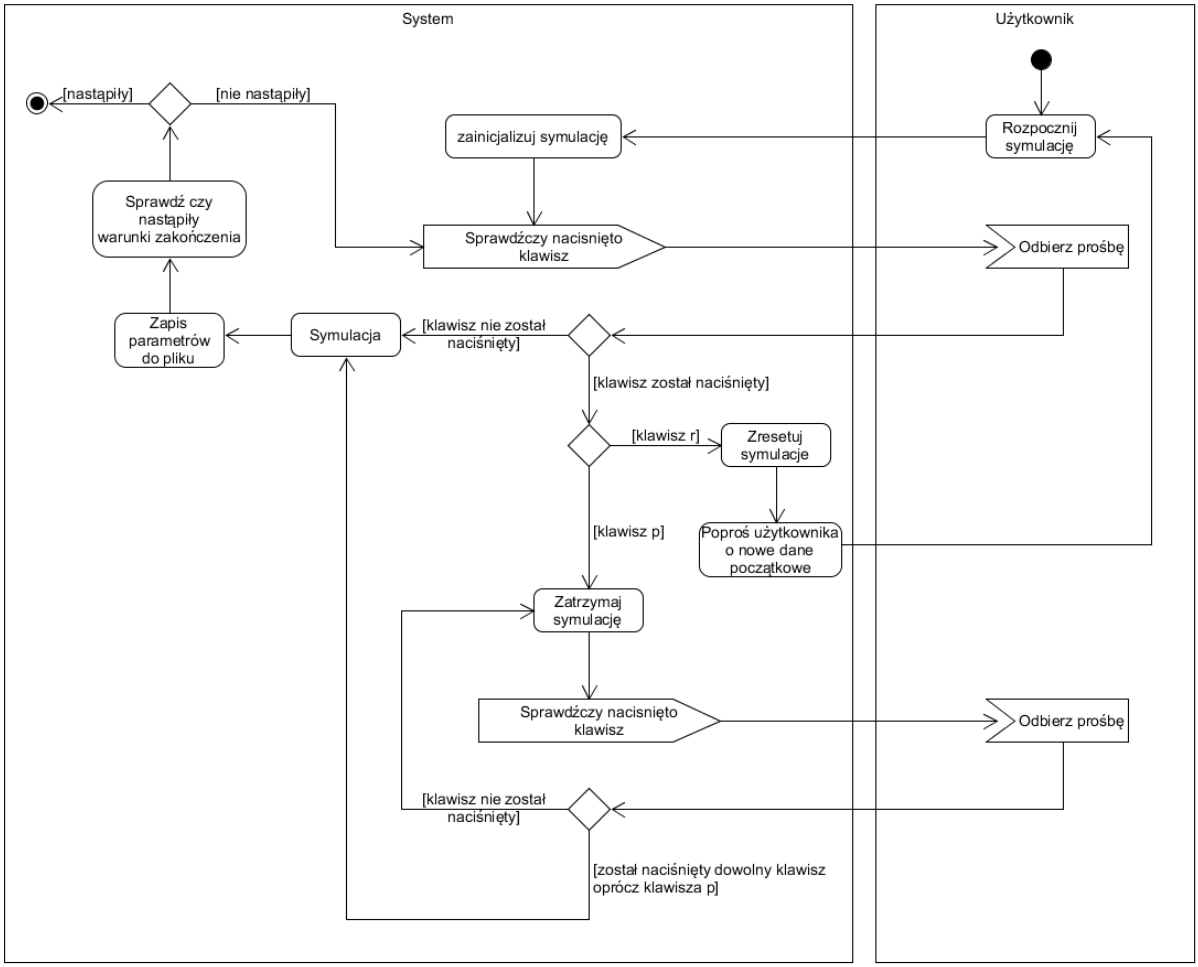
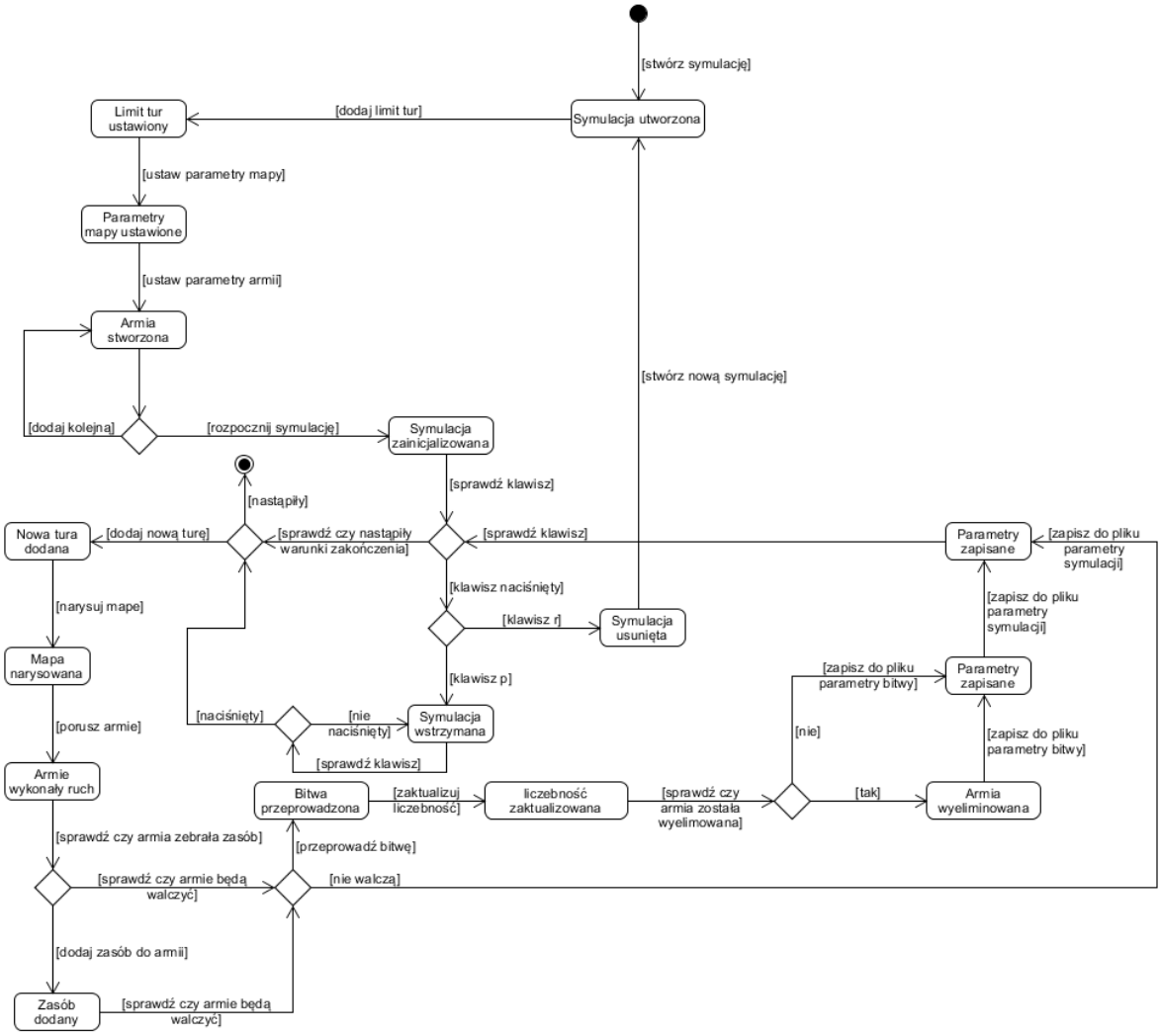

System
Użytkownik
[nastąpiły]
[nie nastąpiły]
zainicjalizuj symulację
Rozpocznij symulację
Sprawdź czy nastąpiły warunki zakończenia
Sprawdźczy nacisnięto klawisz
Odbierz prośbę
Zapis parametrów do pliku
Symulacja
[klawisz nie został naciśnięty]
[klawisz został naciśnięty]
[klawisz r]
Zresetuj symulacje
[klawisz p]
Poproś użytkownika o nowe dane początkowe
Zatrzymaj symulację
Sprawdźczy nacisnięto klawisz
Odbierz prośbę
[klawisz nie został naciśnięty]
[został naciśnięty dowolny klawisz oprócz klawisza p]

Diagram maszyny stanów



Dokumentacja wygenerowana na podstawie komentarzy kodu źródłowego

Armia Class Reference

klasa opisujaca armie - glowny obiekt w symulacji

`#include <Armia.h>`

Public Member Functions

- **Armia** (int, size_t, size_t, char, int, std::string, size_t)
  konstruktor ustawiajacy poczatkowe parametry armii

- **Armia** ()
  konstruktor domyslny

- int **ruch** ()
  funkcja skladowa klasy przeprowadzajaca ruch armii, sprawdzajaca tez czy armia weszla na prowincje gdzie jest inna armia

- size_t **dajPozycjeX** ()
  funkcja skladowa klasy zwracajaca pozycje X

- size_t **dajPozycjeY** ()
  funkcja skladowa klasy zwracajaca pozycje Y

- int **dajPrzyna** ()
  funkcja skladowa klasy zwracajaca przynaleznosc armii

- char **dajSymbol** ()
  funkcja skladowa klasy zwracajaca symbol armii

- std::string **dajNazwe** ()
  funkcja skladowa klasy zwracajaca nazwe armii

- float **dajModyfikatorZasob** ()
  funkcja skladowa klasy zwracajaca modyfikator do obrazen z zebranych zasobow

- bool **czyArmiaAktywna** ()
  funkcja skladowa klasy zwracajaca czy armia jest aktywna

- int **dajLiczebnosc** ()
  funkcja skladowa klasy zwracajaca liczebnosc armii

- int **dajID** ()
  funkcja skladowa klasy zwracajaca ID armii

- void **zmienLiczebnosc** (int a)
  funkcja skladowa klasy zmieniajaca liczebnosc armii

- void **zmienSymbol** (char a)
  funkcja skladowa klasy zmieniajaca symbol armii

- void **zmienAktywnosc** (bool a)
  funkcja skladowa klasy zmieniajaca aktywnosc armii

- void **zbierzZasob** (rodzajeZasobu)
  funkcja skladowa klasy odpowiedzialna za zbieranie zasobu z prowincji

Static Public Member Functions

- static void **inicjalizuj** (size_t, size_t)
  statyczna funkcja skladowa klasy inicjalizujaca granice mapy (potrzebne do tego aby armia nie wyszla poza mape)

Public Attributes

- **Oddzial** * **oddzial**
  skladnik klasy - wskaznik na klase abstrakcyjna ***Oddzial***

- **Artyleria artyleria**
  skladnik klasy - artyleria

- **General general**
  skladnik klasy - general

- **Lekarz lekarz**
  skladnik klasy - lekarz

- **Zwiad zwiad** { **dajPrzyna**() }
  skladnik klasy - zwiad

- **CiezkaJazda ciezkaJazda**
  skladnik klasy - ciezka jazda

- **LekkaJazda lekkaJazda**
  skladnik klasy - lekka jazda

Private Attributes

- bool **aktywna**
  skladnik klasy okreslajacy czy armia jest aktywna

- char **symbol**
  skladnik klasy okreslajacy symbol armii, ktory zostanie wypisany na ekran konsoli na mapie

- int **id**
  skladnik klasy okreslajacy ID armii

- int **liczebnosc**
  skladnik klasy okreslajacy liczbe zolnierzy w armii

- int **przynaleznosc**
  skladnik klasy okreslajacy przynaleznosc danej armii

- size_t **pozycjaX**
  skladnik klasy okreslajacy pozycje x armii na mapie

- size_t **pozycjaY**
  skladnik klasy okreslajacy pozycje y armii na mapie

- float **szybkosc**
  skladnik klasy okreslajacy szybkosc poruszania sie armii

- std::string **nazwaArmii**
  skladnik klasy okreslajacy nazwe armii

- float **modyfikatorObrazenZasob** { 1.0 }
  skladnik klasy okreslajacy o ile procent armia ma zwiekszone zadane obrazenia, np. 1.1 oznacza obraaenia wieksze o 10%

Static Private Attributes

- static size_t **x**
  statyczny skladnik klasy okreslajacy granice x mapy (potrzebne do tego aby armia nie wyszla poza mape)

- static size_t **y**
  statyczny skladnik klasy okreslajacy granice y mapy (potrzebne do tego aby armia nie wyszla poza mape)

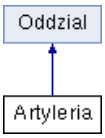
Detailed Description

klasa opisujaca armie - glowny obiekt w symulacji

Artyleria Class Reference

`#include <Artyleria.h>`

Inheritance diagram for Artyleria:



Public Member Functions

- **Artyleria** ()
  konstruktor klasy losujacy wartosc zmiennej modDefensywa

- virtual size_t **dajModyfikator** ()
  funkcja skladowa klasy zwracajaca modOfensywa

Private Attributes

- size_t **modOfensywa**
  skladnik klasy przechowujacy modyfikator do ofensywy armii podczas bitwy

---

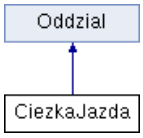
Detailed Description

klasa opisujaca artylerie, jeden z rodzajow oddzialow wojskowych. Obiekt tej klasy dodaje podczas bitwy modyfikator do ofensywy. dziedziczenie publiczne po klasie abstrakcyjnej "Oddzial"

CiezkaJazda Class Reference

`#include <CiezkaJazda.h>`

Inheritance diagram for CiezkaJazda:



Public Member Functions

- **CiezkaJazda** ()
  konstruktor klasy losujacy wartosc zmiennej modObrazenia

- virtual size_t **dajModyfikator** ()
  funkcja skladowa klasy zwracajaca modObrazenia

Private Attributes

- size_t **modObrazenia**
  skladnik klasy przechowujacy modyfikator do obrazen armii podczas bitwy

---

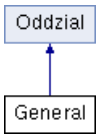
Detailed Description

klasa opisujaca ciezka jazde, jeden z rodzajow oddzialow wojskowych. Obiekt tej klasy dodaje podczas bitwy modyfikator do ofensywy dziedziczenie publiczne po klasie abstrakcyjnej "Oddzial"

General Class Reference

`#include <General.h>`

Inheritance diagram for General:



Public Member Functions

- **General** ()
  konstruktor klasy losujacy wartosc zmiennej modObrazenia

- virtual size_t **dajModyfikator** ()
  funkcja skladowa klasy zwracajaca modObrazenia

Private Attributes

- size_t **modObrazenia**
  skladnik klasy przechowujacy modyfikator do obrazen armii podczas bitwy

---

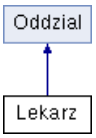
Detailed Description

klasa opisujaca generala, jeden z rodzajow oddzialow wojskowych. Obiekt tej klasy dodaje podczas bitwy modyfikator do obrazen . dziedziczenie publiczne po klasie abstrakcyjnej "Oddzial"

Lekarz Class Reference

`#include <Lekarz.h>`

Inheritance diagram for Lekarz:



Public Member Functions

- **Lekarz** ()
  konstruktor klasy losujacy wartosc zmiennej modLiczebnosc

- virtual size_t **dajModyfikator** ()
  funkcja skladowa klasy zwracajaca modLiczebnosc

Private Attributes

- size_t **modLiczebnosc**
  skladnik klasy przechowujacy modyfikator do liczebnosci armii podczas bitwy

---

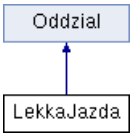
Detailed Description

klasa opisujaca lekarzy, jeden z rodzajow oddzialow wojskowych. Obiekt tej klasy dodaje podczas bitwy modyfikator do liczebnosci armii. dziedziczenie publiczne po klasie abstrakcyjnej "Oddzial"

LekkaJazda Class Reference

`#include <LekkaJazda.h>`

Inheritance diagram for LekkaJazda:



Public Member Functions

- **LekkaJazda** ()
  konstruktor klasy losujacy wartosc zmiennej modDefensywa

- virtual size_t **dajModyfikator** ()
  funkcja skladowa klasy zwracajaca modDefensywa

Private Attributes

- size_t **modDefensywa**
  skladnik klasy przechowujacy modyfikator do defensywy armii podczas bitwy

---

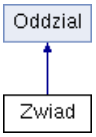
Detailed Description

klasa opisujaca lekka jazde, jeden z rodzajow oddzialow wojskowych. Obiekt tej klasy dodaje podczas bitwy modyfikator do defensywy. dziedziczenie publiczne po klasie abstrakcyjnej "Oddzial"

Zwiad Class Reference

```
#include <Zwiad.h>
```

Inheritance diagram for Zwiad:



Public Member Functions

- **Zwiad** (int)
  konstruktor klasy przypisujacy zwiadowi przynaleznosc taka sama jak armii

- void **raport** ()
  funkcja skladowa klasy, ktora zlicza liczbe zdobytych prowincji przez armie

- virtual size_t **dajModyfikator** ()
  funkcja skladowa klasy zwracajaca modLiczebnosc

Private Attributes

- size_t **modLiczebnosc**
  skladnik klasy przechowujacy modyfikator do liczebnosci armii podczas bitwy, jest on rowny ilosci zdobytych prowincji przez armie

- int **przynaleznosc**
  skladnik klasy przechowujacy przynaleznosc zwiadu

Detailed Description

klasa opisujaca zwiad, jeden z rodzajow oddzialow wojskowych. Obiekt tej klasy dodaje podczas bitwy modyfikator do liczebnosci na podstawie ilosci zdobytych prowincji przez armie

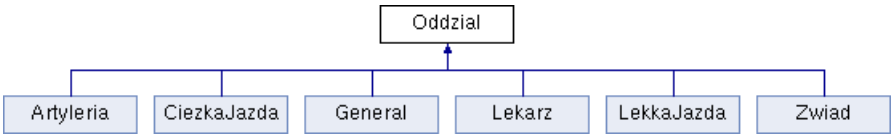
dziedziczenie publiczne po klasie abstrakcyjnej "Oddzial"

Oddzial Class Reference

Klasa abstrakcyjna.

```
#include <Oddzial.h>
```

Inheritance diagram for Oddzial:



Public Member Functions

- virtual size_t **dajModyfikator** ()=0
  funkcja czysto wirtualna, zwraca modyfikator z obiektu na ktory aktualnie jest ustawiony wskaznik do klasy abstrakcyjnej

---

Detailed Description

Klasa abstrakcyjna.

Mapa Class Reference

`#include <Mapa.h>`

Static Public Member Functions

- static std::vector< size_t > **dajLiczbeProwincjiKazdejArmii** (**Armia**, **Armia**)
  statyczna funkcja skladowa klasy zwracajaca liczbe prowincji przejetych dla dwoch armii

- static size_t **dajLiczbeProwincjiArmii** (**Armia**)
  statyczna funkcja skladowa klasy zwracajaca liczbe prowincji jednej armii

- static void **rysuj** (const std::vector< **Armia** > &armie, HANDLE hOut)
  statyczna funkcja skladowa klasy odpowiedzialna za wyswietlenie prowincji oraz armii w konsoli

- static void **inicjalizuj** (size_t, size_t)
  statyczna funkcja skladowa klasy, ktora ustawia pozadany rozmiar tablicy dwywymiarowej

- static void **reset** ()
  statyczna funkcja skladowa klasy, ktora resetuje wszystkie prowincje w tablicy dwuwymiarowej

- static void **clrscr** ()
  statyzcna funkcja skladowa klasy, ktora czysci ekran konsoli po kazdej turze

Static Public Attributes

- static std::vector< std::vector< **Prowincja** > > **mapa**
  statyczny skladnik klasy - tablica dwuwymiarowa prowincji, kazde pole w tej tablicy to unikalna prowincja

- static size_t **dlug**
  statyczny skladnik klasy przechowujacy wielkosc y mapy

- static size_t **szer**
  statyczny skladnik klasy przechowujacy wielkosc x mapy

Detailed Description

Mapa wyglada w nastepujacy sposob:

1. kazde pole na mapie to unikalna prowincja, kolor szary oznacza ze prowincja jest niczyja, jesli prowincja ma kolor inny niz szary - jest ona przejeta przez dana armie
2. armie reprezentowane sa przez znaki 'X', armie poruszaja sie po mapie w sposob calkowicie losowy przy okazji zajmujac prowincje
3. kaada prowincja zawiera jakis zasob, najczesciej jest to zasob "BRAK" czyli zasob nie dajacy zadnych korzysci. Okolo 10% prowincji posiada wartosciowy zasob

klasa odpowiadajaca za mape i jej rysowanie na ekranie konsoli

Operator_bitwy Class Reference

klasa odpowiadajaca za przeprowadzanie bitw pomiedzy armiami

`#include <Operator_bitwy.h>`

Static Public Member Functions

- static size_t **dajLiczbeBitw** ()
  statyczna funkcja skladowa klasy zwracajaca liczbe przeprowadzonych bitw

- static bool **wynik** ()
  statyczna funkcja skladowa klasy losujaca ktora armia bedzie miala status wygranej podczas danej bitwy (straty armii moga jednakze zmienic ten status)

- static std::vector< float > **dajStratyProcentoweArmii** ()
  statyczna funkcja skladowa klasy zwracajaca procentowe straty poniesione przez dwie armie podczas bitwy

- static std::vector< size_t > **bitwa** (**Armia** &, **Armia** &)
  statyczna funckcja skladowa klasy przeprowadzajaca bitwe pomiedzy dwiema armiami

Static Private Attributes

- static size_t **licznikBitw** = 1
  statyczny skladnik klasy przechowujacy liczbe przeprowadzonych bitw

- static float **procentowaStrataArmii1**
  statyczny skladnik klasy przechowujacy procentowa strate armii pierwszej podczas bitwy

- static float **procentowaStrataArmii2**
  statyczny skladnik klasy przechowujacy procentowa strate armii drugiej podczas bitwy

---

Detailed Description

klasa odpowiadajaca za przeprowadzanie bitw pomiedzy armiami

OperatorPliku Class Reference

klasa odpowiadajaca za zapis parametrow symulacji do pliku tekstowego

`#include <OperatorPliku.h>`

Static Public Member Functions

- static bool **zapis** (const std::vector< **Armia** > &armie)
  podstawowa statyczna funkcja skladowa klasy zapisujaca parametry do pliku, zapisuje parametry wszystkich armii na kazda ture

- static bool **zapisPrzedSymulacja** (const std::vector< **Armia** > &)
  statyczna funkcja skladowa klasy zapisujaca do pliku parametry przed rozpoczeciem symulacji

- static bool **zapisPrzedBitwa** (**Armia**, **Armia**)
  statyczna funkcja skladowa klasy zapisujaca do pliku parametry przed rozpoczeciem bitwy

- static bool **zapisBitwy** (**Armia**, **Armia**, std::vector< size_t >, std::vector< float >)
  statyczna funkcja skladowa klasy zapisujaca do pliku przebieg bitwy pomiedzy dwoma armiami

- static bool **ostatniZapis** (**Armia**, ULONGLONG)
  statyczna funkcja skladowa klasy zapisujaca koncowe parametry symulacji (w sytuacji gdy armia wygra poprzez militarne zwyciestwo)

- static bool **ostatniZapisTury** (const std::vector< **Armia** > &, **Armia**, ULONGLONG)
  statyczna funkcja skladowa klasy zapisujaca koncowe parametry symulacji (w sytuacji gdy osiagniety zostanie limit tur)

- static bool **zapisZasobu** (**Armia**, rodzajeZasobu)
  statyczna funkcja skladowa klasy zapisujaca parametry zebranego zasobu przez armie

---

Detailed Description

klasa odpowiadajaca za zapis parametrow symulacji do pliku tekstowego

OperatorSymulacji Class Reference

`#include <OperatorSymulacji.h>`

Public Member Functions

- bool **zainicjalizujSymulacje** ()
  funkcja skladowa klasy inicjalizujaca symulacje wstepnymi parametrami, w niej uzytkownik wpisuje poczatkowe wartosci

- int **prowadzSymulacje** ()
  funkcja skladowa klasy, w ktorej znajduje sie glowna petla symulacji

Private Attributes

- bool **czySymulacjaJestAktywna**
  skladnik klasy przechowujacy stan symulacji

- std::vector< **Armia** > **armie**
  skladnik klasy - tablica przechowujaca armie

- std::vector< size_t > **zmianaLiczebnosci**
  skladnik klasy - tablica przechowujaca zmiany liczebnosci armii po bitwie

- **Armia armiaZwycieska**
  skladnik klasy - obiekt do ktorego zostaje wpisana armia, ktora wygrala (w przypadku zwyciestwa militarnego)

- ULONGLONG **czasTrwaniaSymulacji** {}
  skladnik klasy przechowujacy czas trwania symulacji

- ULONGLONG **czasTrwaniaPrzerwy** {}
  skladnik klasy przechowujacy czas trwania przerwy podczas symulacji

- HANDLE **hOut**
  skladnik klasy - uchwyt na wyjscie konsoli

- const size_t **minimum** { 10 }
  skladnik klasy przechowujacy minimalny dozwolony rozmiar mapy

- const size_t **maksimumX** { 40 }
  skladnik klasy przechowujacy maksymalne dozwolone rozmiary mapy x

- const size_t **maksimumY** { 40 }
  skladnik klasy przechowujacy maksymalne dozwolone rozmiary mapy y

---

Detailed Description

uzytkownik podaje z klawiatury nastepujace dane poczatkowe:

- maksymalny limit tur
- rozmiar mapy
- parametry kazdej z armii: pozycja poczatkowa x, pozycja poczatkowa y, liczebnosc, nazwa
- minimalnie sa 2 armie, uzytkownik moze dodac ich maksymalnie 10

klasa odpowiadajaca za przeprowadzenie symulacji

OperatorZakonczenia Class Reference

`#include <OperatorZakonczenia.h>`

Static Public Member Functions

- static void **ustawMaksymalnyLimitTur** (size_t)
  statyczna funkcja skladowa klasy ustawiajaca maksymalny limit tur

- static bool **sprawdzCzySymulacjaSieZakonczylaArmie** (const std::vector< **Armia** > &)
  statyczna funkcja skladowa klasy sprawdzajaca czy symulacja zakonczyla sie zwyciestwem militarnym ktorejs z armii

- static bool **sprawdzCzySymulacjaSieZakonczylaTury** ()
  statyczna funkcja skladowa klasy sprawdzajaca czy symulacja zakonczyla sie przez osiagniecie limitu tur

- static std::vector< size_t > **podajLiczbeProwincjiArmii** (const std::vector< **Armia** > &)
  statyczna funkcja skladowa klasy zliczajaca ile kazda armia posiada przejetych prowincji

- static **Armia pokazArmieZwycieska** (const std::vector< **Armia** > &)
  statyczna funkcja skladowa klasy wyznaczajaca wygrana armie

- static size_t **dajMaxTure** ()
  statyczna funkcja skladowa klasy zwracajaca maksymalna liczbe tur

Static Private Attributes

- static size_t **maxTura** = 3000
  skladnik klasy przechowujacy maksymalna liczbe tur

Detailed Description

istnieja dwa mozliwe zakonczenia symulacji:

1. zwyciestwo militarne - w przypadku gdy na polu walki zostanie tylko jedna armia - wygrywa ona symulacje
2. zakonczenie przez osiagniecie limtu tur - w przypadku gdy w symulacji zostanie osiagniety ustalony limit tur a co najmniej dwie armie zyja - wygrywa armia, ktora przejela wiecej prowincji

klasa odpowiadajaca za sprawdzenie czy wystapily warunki zakonczenia symulacji

Prowincja Class Reference

klasa odopowiadajaca za prowincje

`#include <Prowincja.h>`

Public Member Functions

- **Prowincja** ()
  konstruktor klasy ustawiajacy poczatkowe wartosci opisujace prowincje

- int **dajPrzynaleznosc** ()
  funkcja skladowa klasy zwracajaca przynaleznsoc prowincji

- char **dajSymbol** ()
  funkcja skladowa klasy zwracajaca symbol prowincji

- int **dajArmieWProwincji** ()
  funkcja skladowa klasy zwracajaca armie w prowincji

- int **dajwspolrzednax** ()
  funkcja skladowa klasy zwracajaca wspolrzedna x prowincji

- int **dajwspolrzednay** ()
  funkcja skladowa klasy zwracajaca wspolrzedna y prowincji

- void **zmienPrzynaleznosc** (int a)
  funkcja skladowa klasy zmieniajaca przynaleznosc

- void **zmienSymbol** (char a)
  funkcja skladowa klasy zmieniajaca symbol

- void **zmienArmieWProwincji** (int a)
  funkcja skladowa klasy zmieniajaca armie w prowincji

- void **zmienwspolrzednax** (int a)
  funkcja skladowa klasy zmieniajaca wspolrzedna x prowincji

- void **zmienwspolrzednay** (int a)
  funkcja skladowa klasy zmieniajaca wspolrzedna y prowincji

Public Attributes

- **Zasob zasobProwincji**
  skladnik klasy - zasob ktory posiada prowincja

Private Attributes

- int **przynaleznosc** {}
  skladnik klasy oznaczajacy do jakiej armii nalezy dana prowincja

- char **symbol** {}
  skladnik klasy przechowujacy symbol wyswietlany na ekranie konsoli

- int **wspolrzednax**
  skladnik klasy przechowujacy wspolrzedna x prowincji

- int **wspolrzednay**
  skladnik klasy przechowujacy wspolrzedna y prowincji

- int **armia_w_prowincji** {}
  skladnik klasy przechowujacy id armii w prowincji

Detailed Description

klasa odopowiadajaca za prowincje

Zasob Class Reference

```
#include <Zasob.h>
```

Public Member Functions

- **Zasob** ()
  konstruktor klasy losujaca rodzaj zasobu

- rodzajeZasobu **dajRodzajZasobu** ()
  funkcja skladowa klasy zwracajacy rodzaj zasoby ktory przechowuje obiekt tej klasy

- void **zmienAktywnoscZasobuNa** (bool pom)
  funkcja skladowa klasy zmieniajaca aktywnosc zasobu na wskazany

- bool **dajAktywnoscZasobu** ()
  funkcja skladowa klasy zwracajaca aktywnosc zasobu

Private Attributes

- rodzajeZasobu **rodzajZasobu**
  skladnik klasy przechowujacy rodzaj zasobu

- bool **aktywnoscZasobu**
  skladnik klasy przechowujacy czy dany zasob jest aktywny

Zegar Class Reference

`#include <Zegar.h>`

Static Public Member Functions

- static void **start** ()
  statyczna funkcja skladowa klasy ustawiajaca ture na 0

- static void **nowaTura** ()
  statyczna funkcja skladowa klasy zwiekszajaca ture o 1

- static size_t **dajObecnaTure** ()
  statyczna funkcja skladowa klasy zwracajaca liczbe tur

- static void **zacznijOdmierzacCzas** ()
  statyczna funkcja skladowa klasy zaczynajaca mierzyc czas rzeczywisty

- static ULONGLONG **dajCzasWykonaniaSymulacjiMiliSekundy** ()
  statyczna funkcja skladowa klasy, ktora zwraca czas pomiedzy jej wywolaniem a wywolaniem funkcji ***zacznijOdmierzacCzas()***

Static Private Attributes

- static size_t **tura** = 0
  statyczny skladnik klasy przechowujacy liczbe tur

- static ULONGLONG **czasStartu**
  statyczny skladnik klasy przechowujacy czas startu symulacji

- static ULONGLONG **czasKonca**
  statyczny skladnik klasy przechowujacy czas startu symulacji

---

Detailed Description

zliczanie czasu rzeczywistego odbywa sie za pomoca funkcji GetTickCount64() z biblioteki Windows.h
klasa odpowiedzialna za mierzenie uplywu czasu i tur