

# **Dokumentacja projektu – programowanie obiektowe**

**Autorzy:**

**Kamil Gawlik**

**Sebastian Dorabiała**

## Spis treści

Opis projektu.....	5
Analiza czasownikowo-rzeczownikowa.....	6
Karty CRC .....	8
Diagramy przypadków użycia .....	11
Diagram klas .....	12
Diagram obiektów .....	13
Diagramy sekwencji.....	14
Diagramy aktywności .....	16
Diagram maszyny stanów .....	18
Dokumentacja wygenerowana na podstawie komentarzy kodu źródłowego .....	19
Armia Class Reference .....	19
Public Member Functions .....	19
Static Public Member Functions .....	20
Public Attributes.....	20
Private Attributes .....	20
Static Private Attributes .....	21
Detailed Description.....	21
Artyleria Class Reference.....	22
Public Member Functions .....	22
Private Attributes .....	22
Detailed Description.....	22
CiezkaJazda Class Reference .....	22
Public Member Functions .....	22
Private Attributes .....	22
Detailed Description.....	23
General Class Reference.....	23
Public Member Functions .....	23
Private Attributes .....	23
Detailed Description.....	23
Lekarz Class Reference .....	24
Public Member Functions .....	24

Private Attributes .....	24
Detailed Description.....	24
LekkaJazda Class Reference .....	24
Public Member Functions .....	24
Private Attributes .....	24
Detailed Description.....	25
Zwiad Class Reference .....	25
Public Member Functions .....	25
Private Attributes .....	25
Detailed Description.....	25
Oddzial Class Reference.....	25
Public Member Functions .....	26
Detailed Description.....	26
Mapa Class Reference .....	27
Static Public Member Functions .....	27
Static Public Attributes .....	27
Detailed Description.....	27
Operator_bitwy Class Reference .....	28
Static Public Member Functions .....	28
Static Private Attributes .....	28
Detailed Description.....	28
OperatorPliku Class Reference.....	29
Static Public Member Functions .....	29
Detailed Description.....	29
OperatorSymulacji Class Reference .....	30
Public Member Functions .....	30
Private Attributes .....	30
Detailed Description.....	30
OperatorZakonczenia Class Reference.....	31
Static Public Member Functions .....	31
Static Private Attributes .....	31
Detailed Description.....	31
Prowincja Class Reference .....	32

Public Member Functions .....	32
Public Attributes.....	32
Private Attributes .....	32
Detailed Description.....	33
Zasob Class Reference .....	33
Public Member Functions .....	33
Private Attributes .....	33
Zegar Class Reference .....	34
Static Public Member Functions .....	34
Static Private Attributes .....	34
Detailed Description.....	34

## 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 możliwość poruszania armią na mapie, oraz zbierania zasobów z mapy. Armie 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 pożądanых 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 armie 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 Oddział opisujący Artylerię. Obiekt będzie przekazywać modyfikator do ofensywy podczas bitwy.

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

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

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

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

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

## Karty CRC

Armia	
<ul style="list-style-type: none"> <li>• Poruszanie armii po mapie</li> <li>• Zbieranie zasobów z mapy</li> </ul>	<ul style="list-style-type: none"> <li>• Oddział</li> <li>• Artyleria</li> <li>• General</li> <li>• Lekka jazda</li> <li>• Ciezka jazda</li> <li>• Lekarz</li> <li>• Zwiad</li> <li>• Mapa</li> <li>• rodzaje zasobu</li> <li>• Zasob</li> <li>• Prowincja</li> </ul>

Abstract	<b>Oddział</b> Artyleria, General, Lekka jazda, Ciezka jazda, Lekarz, Zwiad
<ul style="list-style-type: none"> <li>• Dodawanie armii modyfikatorów podczas bitwy</li> </ul>	

Artyleria		Oddział
<ul style="list-style-type: none"> <li>• Dodawanie armii modyfikatora do ofensywy podczas bitwy</li> </ul>	<ul style="list-style-type: none"> <li>• Armia</li> </ul>	

Ciezka jazda		Oddział
<ul style="list-style-type: none"> <li>• Dodawanie armii modyfikatora do obrażeń podczas bitwy</li> </ul>	<ul style="list-style-type: none"> <li>• Armia</li> </ul>	

General		Oddział
<ul style="list-style-type: none"> <li>• Dodawanie armii modyfikatora do obrażeń podczas bitwy</li> </ul>	<ul style="list-style-type: none"> <li>• Armia</li> </ul>	

Lekarz		Oddział
<ul style="list-style-type: none"> <li>• Dodawanie armii modyfikatora do liczebności podczas bitwy</li> </ul>	<ul style="list-style-type: none"> <li>• Armia</li> </ul>	

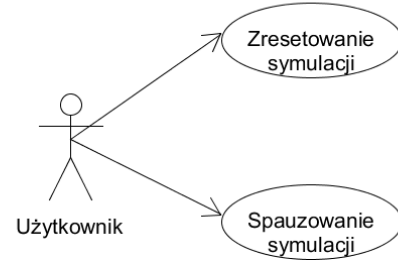
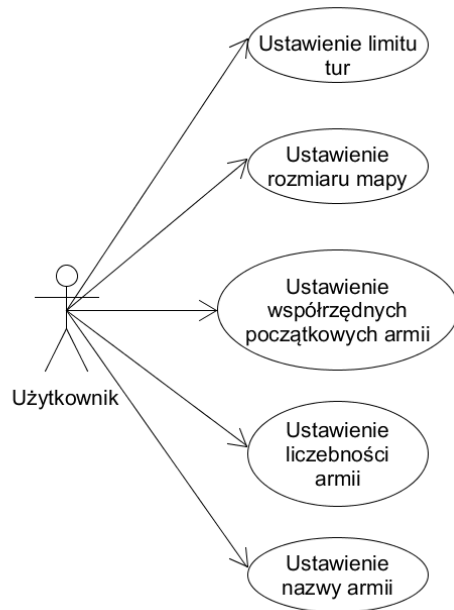
Lekka jazda		Oddział
<ul style="list-style-type: none"> <li>• Dodawanie armii modyfikatora do defensywy podczas bitwy</li> </ul>	<ul style="list-style-type: none"> <li>• Armia</li> </ul>	



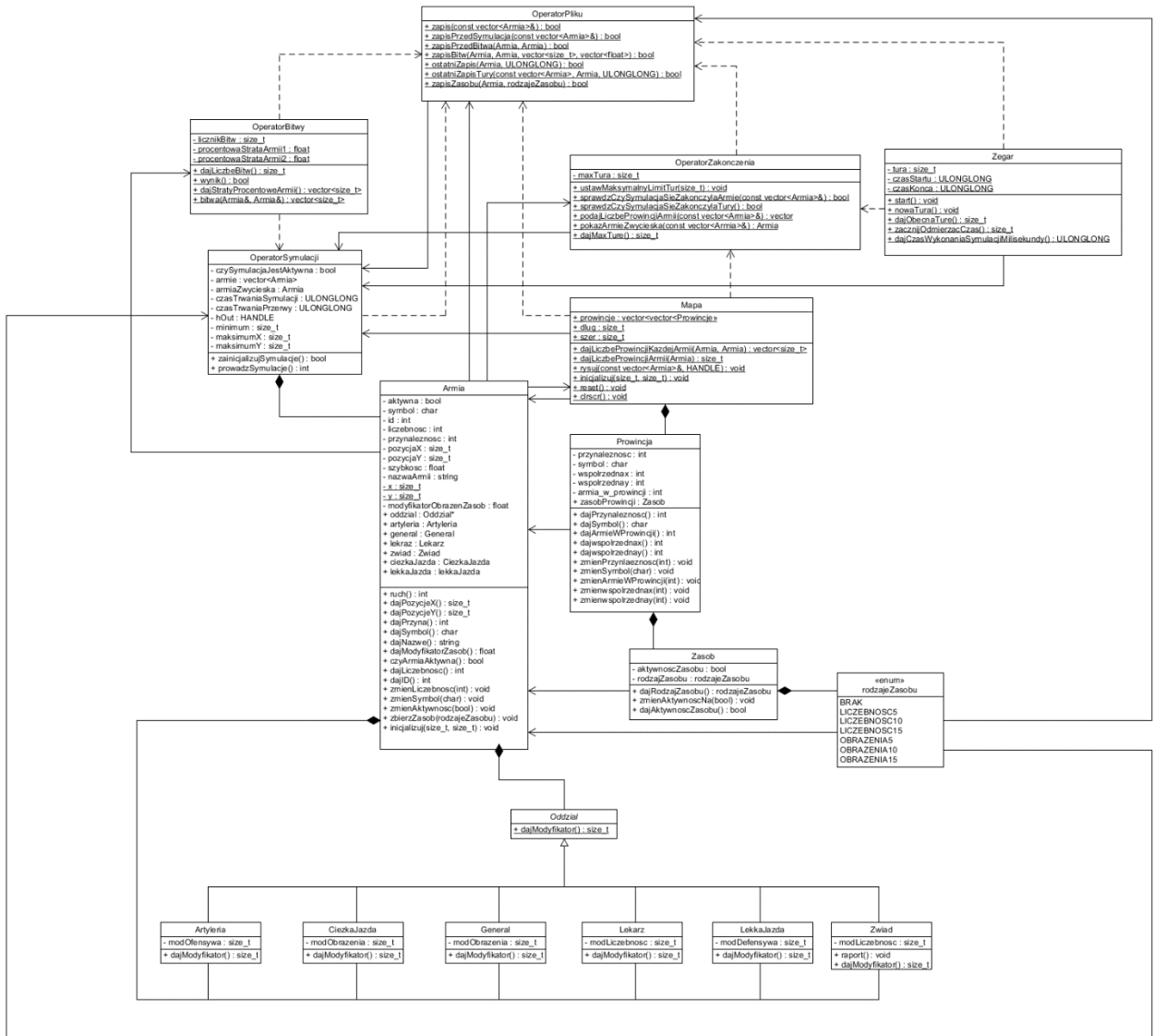
Zwiad		Oddzial
<ul style="list-style-type: none"> <li>• Dodawanie armii modyfikatora do liczebności podczas bitwy</li> <li>• Przeprowadzanie rozpoznania dla armii przed bitwą</li> </ul>		<ul style="list-style-type: none"> <li>• Armia</li> </ul>
Operator_bitwy		
<ul style="list-style-type: none"> <li>• Przeprowadzanie bitw pomiędzy dwiema armiami</li> <li>• Wyliczenie strat w liczebności armii biorących udział w bitwie</li> <li>• Wyznaczanie wygranego i przegranego danej bitwy</li> <li>• Liczenie ilości przeprowadzonych bitw</li> </ul>		<ul style="list-style-type: none"> <li>• Armia</li> </ul>
OperatorPliku		
<ul style="list-style-type: none"> <li>• Zapisywanie do pliku wszystkich parametrów symulacji</li> </ul>		<ul style="list-style-type: none"> <li>• Armia</li> <li>• rodzajeZasobu</li> <li>• Zegar</li> <li>• OperatorZakonczenia</li> <li>• Mapa</li> <li>• OperatorBitwy</li> <li>• OperatorSymulacji</li> </ul>
OperatorSymulacji		
<ul style="list-style-type: none"> <li>• Wczytanie początkowych parametrów od użytkownika</li> <li>• Sprawdzenie poprawności wprowadzonych danych</li> <li>• Przeprowadzenie symulacji</li> </ul>		<ul style="list-style-type: none"> <li>• Armia</li> <li>• Mapa</li> <li>• Zegar</li> <li>• OperatorZakonczenia</li> <li>• OperatorPliku</li> <li>• rodzajeZasobu</li> <li>• Operator_bitwy</li> </ul>
OperatorZakonczenia		
<ul style="list-style-type: none"> <li>• Kontrolowanie czy nastąpiły warunki zakończenia symulacji</li> <li>• Przechowywanie maksymalnej liczby tur</li> <li>• Wybieranie zwycięzcy symulacji</li> </ul>		<ul style="list-style-type: none"> <li>• Armia</li> <li>• Mapa</li> <li>• Zegar</li> </ul>
Prowincja		
<ul style="list-style-type: none"> <li>• Przechowywanie danych o prowincji</li> </ul>		<ul style="list-style-type: none"> <li>• Zasob</li> </ul>

Zasob	
<ul style="list-style-type: none"><li>• Losowanie rodzaju zasobu</li><li>• Kontrolowanie aktywności zasobu</li></ul>	<ul style="list-style-type: none"><li>• rodzajeZasobu</li></ul>
Mapa	
<ul style="list-style-type: none"><li>• Rysowanie na ekranie konsoli mapy (prowincji) i armii</li><li>• Czyszczenie ekranu po każdym przebiegu symulacji</li><li>• Aktualizacja mapy po każdym przebiegu symulacji</li><li>• Przechowywanie informacji o rozmiarze mapy</li></ul>	<ul style="list-style-type: none"><li>• Armia</li><li>• Prowincja</li></ul>
Zegar	
<ul style="list-style-type: none"><li>• Kontrola i zmiana liczby tur</li><li>• Zliczanie rzeczywistego czasu trwania symulacji</li></ul>	

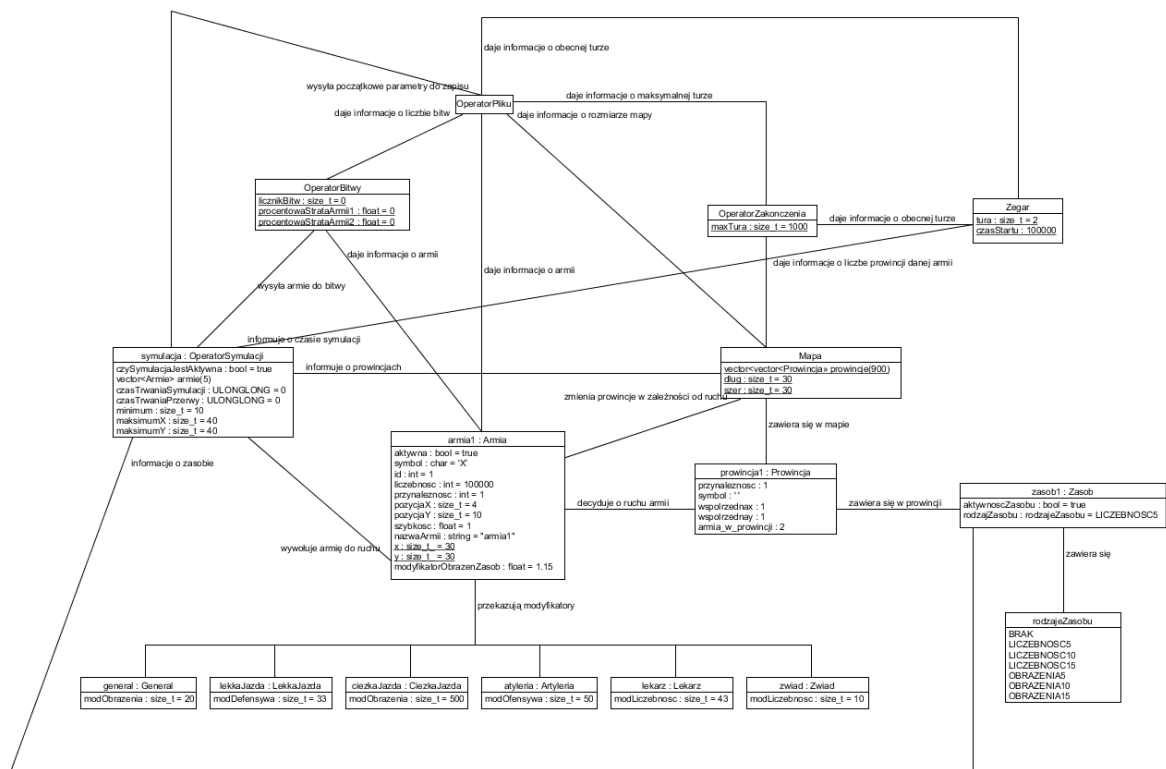
## Diagramy przypadków użycia



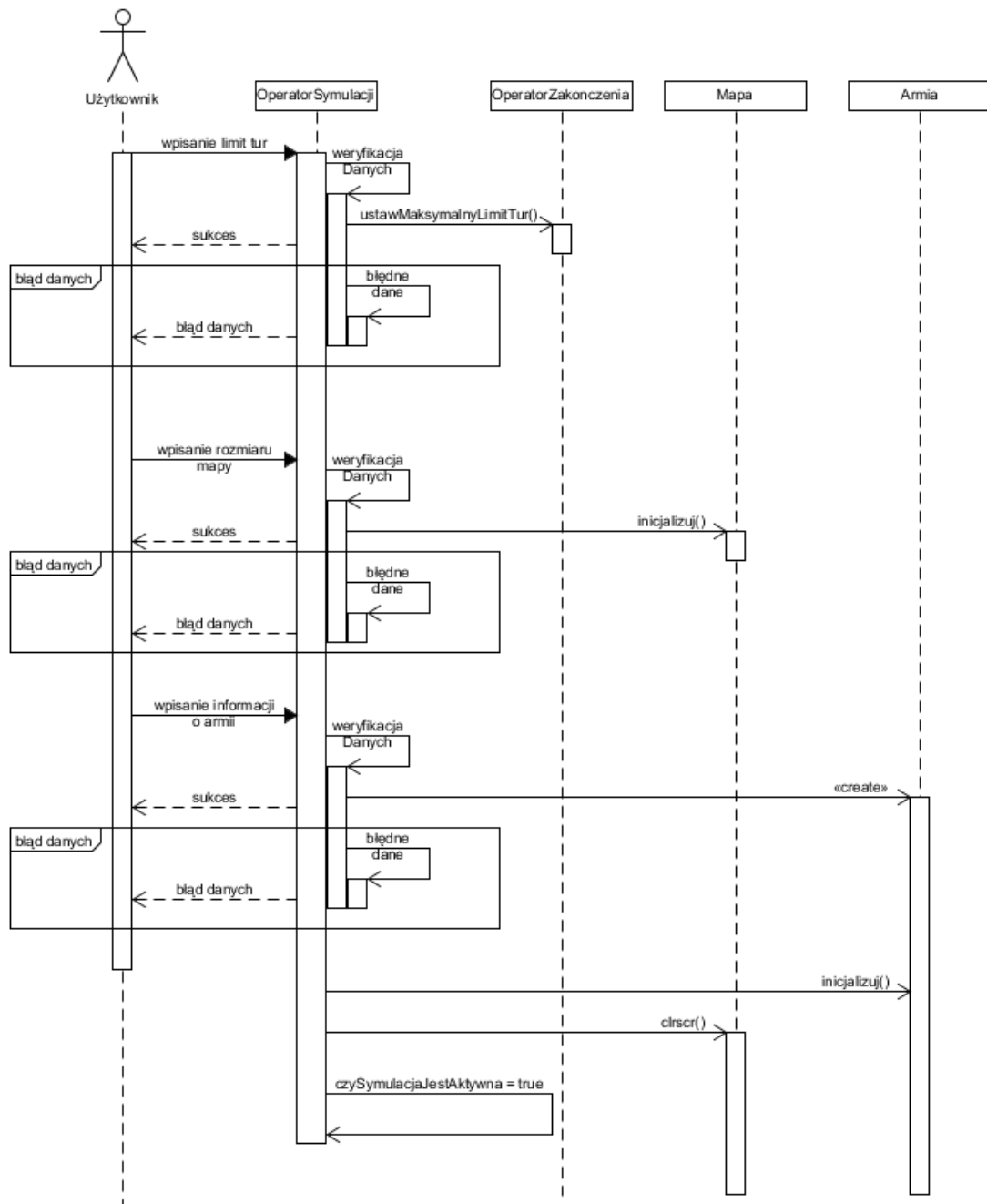
# Diagram klas

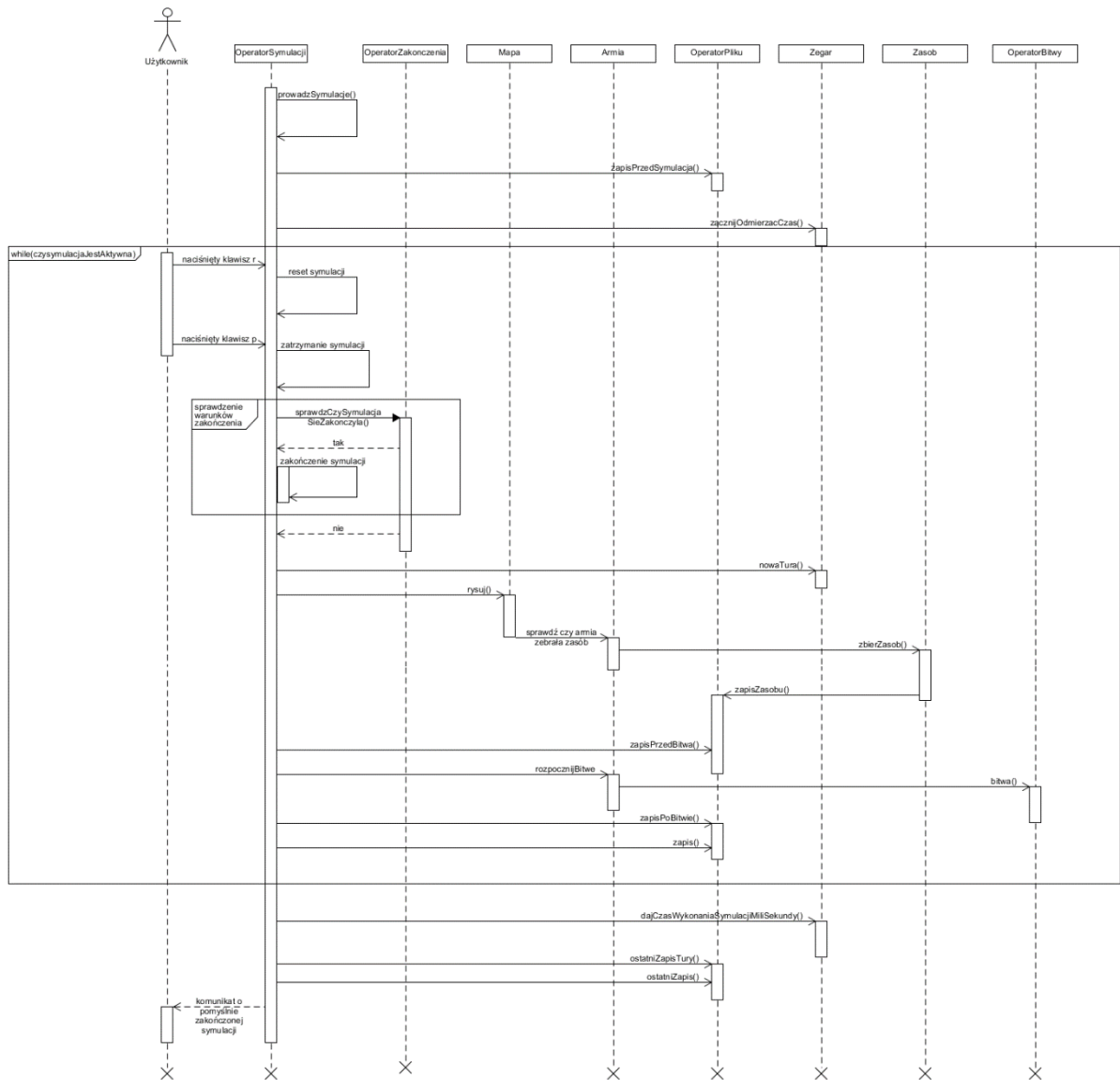


# Diagram obiektów

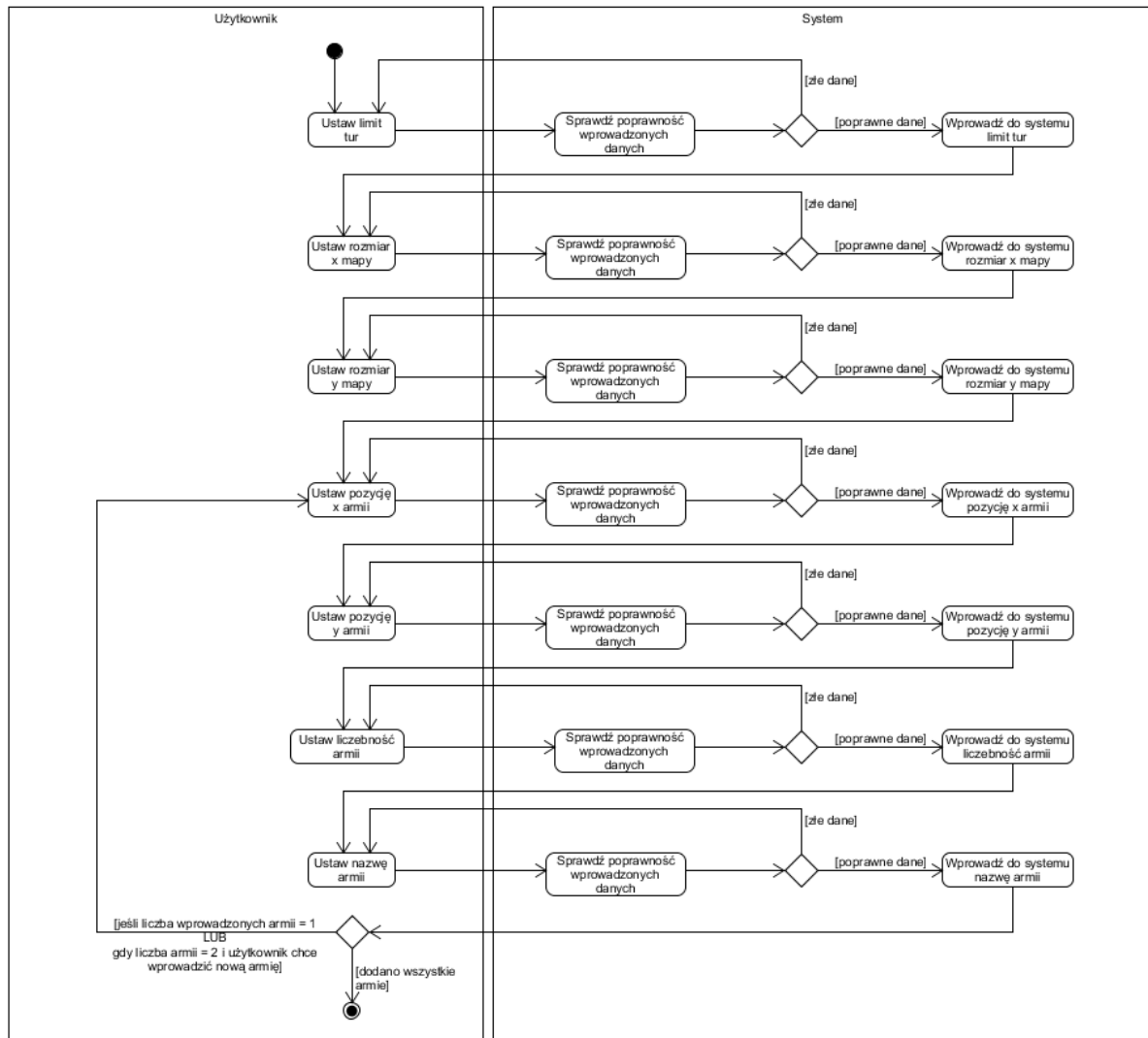


# Diagramy sekwencji

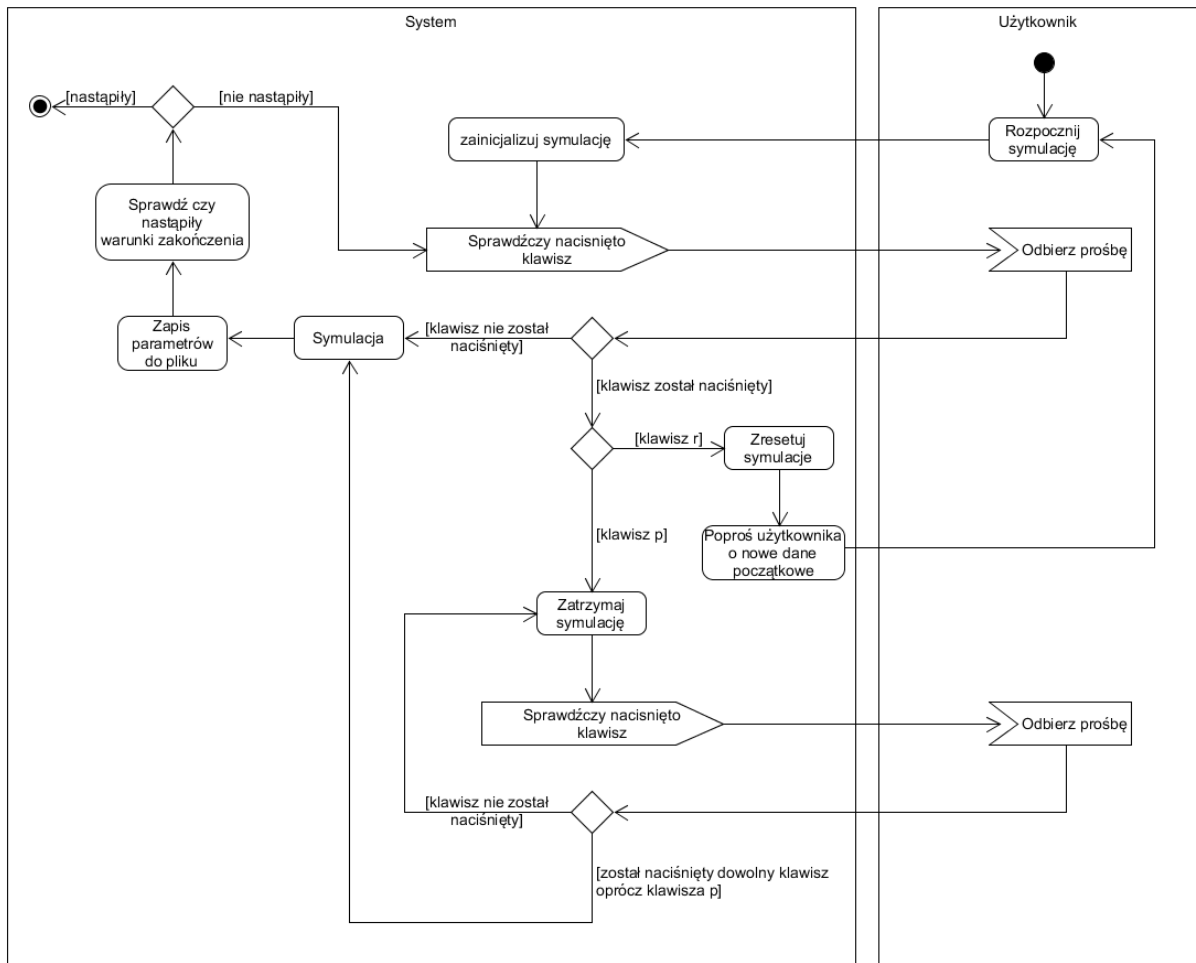




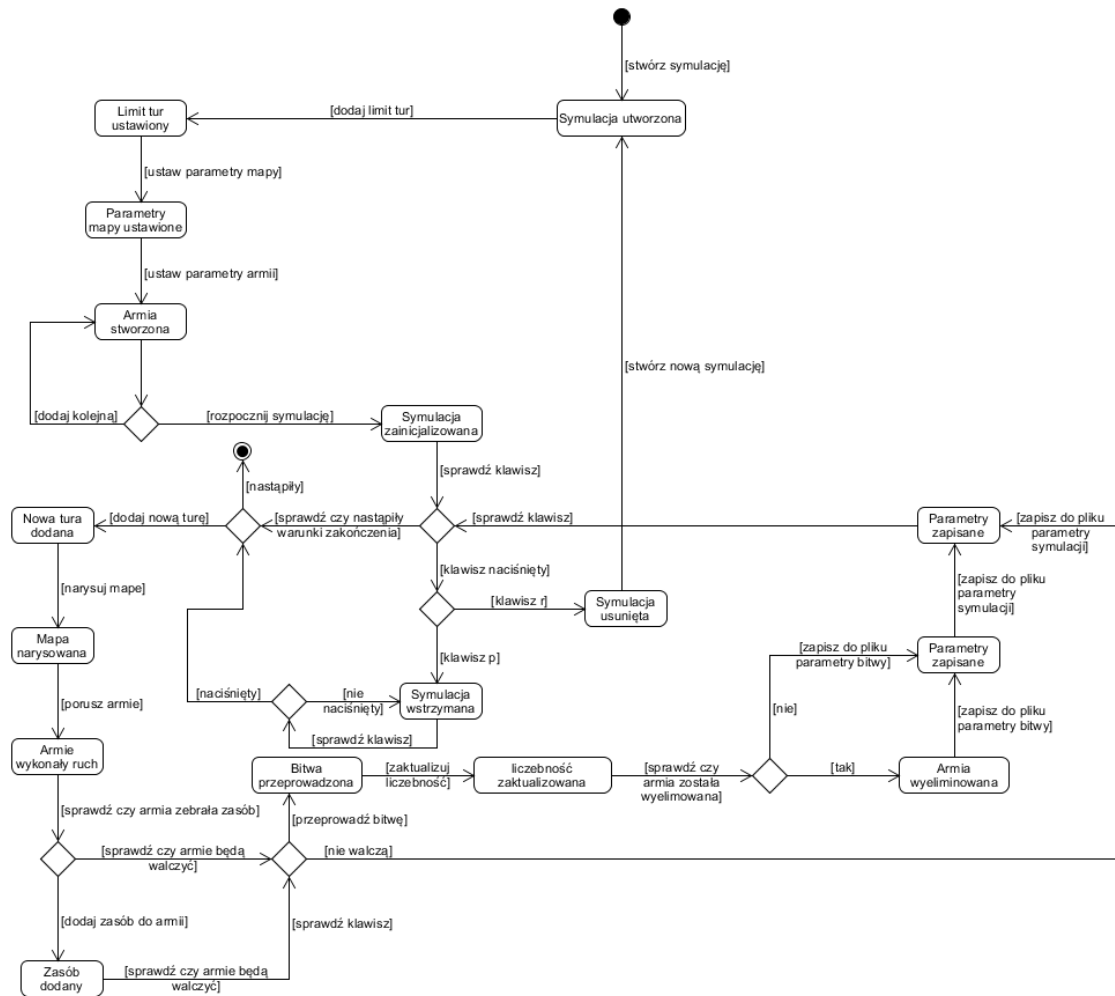
# Diagramy aktywności







# Diagram maszyny stanów



# Dokumentacja wygenerowana na podstawie komentarzy kodu źródłowego

## Armia Class Reference

klasa opisująca armie - główny obiekt w symulacji

```
#include <Armia.h>
```

### Public Member Functions

- **Armia** (int, size\_t, size\_t, char, int, std::string, size\_t)  
*konstruktor ustawiający początkowe parametry armii*
- **Armia** ()  
*konstruktor domyslny*
- int **ruch** ()  
*funkcja składowa klasy przeprowadzająca ruch armii, sprawdzająca też czy armia weszła na prowincję gdzie jest inna armia*
- size\_t **dajPozycjeX** ()  
*funkcja składowa klasy zwracająca pozycję X*
- size\_t **dajPozycjeY** ()  
*funkcja składowa klasy zwracająca pozycję Y*
- int **dajPrzyna** ()  
*funkcja składowa klasy zwracająca przynależność armii*
- char **dajSymbol** ()  
*funkcja składowa klasy zwracająca symbol armii*
- std::string **dajNazwe** ()  
*funkcja składowa klasy zwracająca nazwę armii*
- float **dajModyfikatorZasob** ()  
*funkcja składowa klasy zwracająca modyfikator do obrazów z zebranych zasobów*
- bool **czyArmiaAktywna** ()  
*funkcja składowa klasy zwracająca czy armia jest aktywna*
- int **dajLiczebnosc** ()  
*funkcja składowa klasy zwracająca liczebność armii*
- int **dajID** ()  
*funkcja składowa klasy zwracająca ID armii*

- void **zmienLiczebnosc** (int a)  
*funkcja skladowa klasy zmieniajaca liczebosc 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 liczebność**  
*składnik klasy określający liczbę żołnierzy w armii*
- **int przynależność**  
*składnik klasy określający przynależność danej armii*
- **size\_t pozycjaX**  
*składnik klasy określający pozycję x armii na mapie*
- **size\_t pozycjaY**  
*składnik klasy określający pozycję y armii na mapie*
- **float szybkość**  
*składnik klasy określający szybkość poruszania się armii*
- **std::string nazwaArmii**  
*składnik klasy określający nazwę armii*
- **float modyfikatorObrazenZasob { 1.0 }**  
*składnik klasy określający o ile procent armia ma zwiększone zadane obrażenia, np. 1.1 oznacza obrażenia większe o 10%*

#### Static Private Attributes

- **static size\_t x**  
*statyczny składnik klasy określający granicę x mapy (potrzebne do tego aby armia nie wyszła poza mapę)*
- **static size\_t y**  
*statyczny składnik klasy określający granicę y mapy (potrzebne do tego aby armia nie wyszła poza mapę)*

---

#### Detailed Description

klasa opisująca armie - główny 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 składowa klasy zwracajaca modOfensywa*

### Private Attributes

- size\_t **modOfensywa**  
*skladnik klasy przechowujacy modyfikator do ofensywy armii podczas bitwy*

---

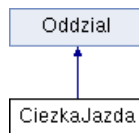
### Detailed Description

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 składowa klasy zwracajaca modObrazenia*

### Private Attributes

- size\_t **modObrazenia**  
*skladnik klasy przechowujacy modyfikator do obrazow 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 obrazem armii podczas bitwy*

---

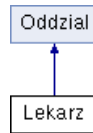
### Detailed Description

klasa opisujaca generala, jeden z rodzajow oddzialow wojskowych. Obiekt tej klasy dodaje podczas bitwy modyfikator do obrazem . 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 modLiczebosc*
- virtual size\_t **dajModyfikator ()**  
*funkcja składowa klasy zwracajaca modLiczebosc*

### Private Attributes

- size\_t **modLiczebosc**  
*składnik klasy przechowujący modyfikator do liczebności armii podczas bitwy*

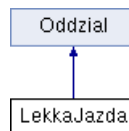
### Detailed Description

klasa opisująca lekarzy, jeden z rodzajów oddziałów wojskowych. Obiekt tej klasy dodaje podczas bitwy modyfikator do liczebności 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 składowa klasy zwracajaca modDefensywa*

### Private Attributes

- size\_t **modDefensywa**  
*składnik klasy przechowujący 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 składowa klasy, która zlicza liczbę zdobytych prowincji przez armie*
- virtual size\_t **dajModyfikator** ()  
*funkcja składowa klasy zwracająca modLiczebność*

### Private Attributes

- size\_t **modLiczebność**  
*składnik klasy przechowujący modyfikator do liczebności armii podczas bitwy, jest on równy ilości zdobytych prowincji przez armie*
- int **przynaleznosc**  
*składnik klasy przechowujący przynaleznosc zwiadu*

---

### Detailed Description

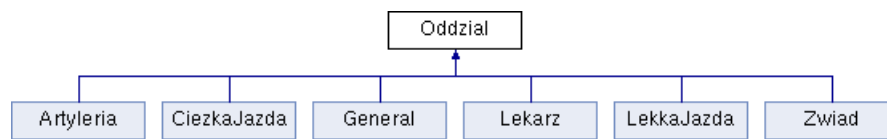
klasa opisujaca zwiad, jeden z rodzajow oddzialow wojskowych. Obiekt tej klasy dodaje podczas bitwy modyfikator do liczebności na podstawie ilości zdobytych prowincji przez armie  
dziedziczenie publiczne po klasie abstrakcyjnej "Oddzial"

### 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 który aktualnie jest ustawiony wskaźnik 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 dwuwymiarowej*
- static void **reset** ()  
*statyczna funkcja skladowa klasy, ktora resetuje wszystkie prowincje w tablicy dwuwymiarowej*
- static void **clrscr** ()  
*statyczna 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 arme
2. arme reprezentowane sa przez znaki 'X', arme 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 funkcja 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 osiagniete 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 składowa klasy inicjalizująca symulację wstępnymi parametrami, w niej użytkownik wpisuje początkowe wartości*
- **int prowadzSymulacje ()**  
*funkcja składowa klasy, w której znajduje się główna pętla symulacji*

### Private Attributes

- **bool czySymulacjaJestAktywna**  
*składnik klasy przechowujący stan symulacji*
- **std::vector< Armia > armie**  
*składnik klasy - tablica przechowująca armie*
- **std::vector< size\_t > zmianaLiczebnosci**  
*składnik klasy - tablica przechowująca zmiany liczebności armii po bitwie*
- **Armia armiaZwycieska**  
*składnik klasy - obiekt do którego zostaje wpisana armia, która wygrała (w przypadku zwycięstwa militarnego)*
- **ULONGLONG czasTrwaniaSymulacji {}**  
*składnik klasy przechowujący czas trwania symulacji*
- **ULONGLONG czasTrwaniaPrzerwy {}**  
*składnik klasy przechowujący czas trwania przerwy podczas symulacji*
- **HANDLE hOut**  
*składnik klasy - uchwyt na wyjście konsoli*
- **const size\_t minimum { 10 }**  
*składnik klasy przechowujący minimalny dozwolony rozmiar mapy*
- **const size\_t maksimumX { 40 }**  
*składnik klasy przechowujący maksymalne dozwolone rozmiary mapy x*
- **const size\_t maksimumY { 40 }**  
*składnik klasy przechowujący maksymalne dozwolone rozmiary mapy y*

---

### Detailed Description

użytkownik podaje z klawiatury następujące dane początkowe:

- maksymalny limit tur
- rozmiar mapy
- parametry kazdej z armii: pozycja poczatkowa x, pozycja poczatkowa y, liczebność, 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 **sprawdzCzySymulacjaSieZakonczyłaArmie** (const std::vector< **Armia** > &)  
*statyczna funkcja skladowa klasy sprawdzajaca czy symulacja zakonczyla sie zwyciestwem militarnym ktorejs z armii*
- static bool **sprawdzCzySymulacjaSieZakonczyłaTury** ()  
*statyczna funkcja skladowa klasy sprawdzajaca czy symulacja zakonczyla sie przez osiagniecie limitu tur*
- static std::vector< size\_t > **podajLiczeProwincjiArmii** (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 limitu tur - w przypadku gdy w symulacji zostanie osiagniety ustalony limit tur a co najmniej dwie armie zyja - wygrywa armia, ktora przeejala wiecej prowincji

klasa odpowiadajaca za sprawdzenie czy wystapily warunki zakonczenia symulacji

## Prowincja Class Reference

klasa odpowiadajaca za prowincje

```
#include <Prowincja.h>
```

### Public Member Functions

- **Prowincja ()**  
*konstruktor klasy ustawiajacy poczatkowe wartosci opisujace prowincje*
- **int dajPrzynaleznosc ()**  
*funkcja skladowa klasy zwracajaca przynaleznosc 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 wspolrzeczna x prowincji*
- **int wspolrzednay**  
*skladnik klasy przechowujacy wspolrzeczna y prowincji*
- **int armia\_w\_prowincji { }**  
*skladnik klasy przechowujacy id armii w prowincji*

## Detailed Description

klasa odpowiadajaca 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