10.06.2018 Wrocław

# Programowanie Obiektowe

INEW0003P

Projekt

| | |
|---|---|
| **Wydział Elektroniki** | **Kierunek:** Informatyka |
| **Grupa zajęciowa**: wt. $17^{05}$-$18^{45}$ | **Semestr:** 2017 / 2018 lato |
| **Nazwisko i Imię**: Szymon Wiśniewski | **Nr indeksu**: 2[redacted] |
| **Nr grupy projektowej**: E00-93az | |
| **Prowadzący**: mgr inż. Jakub Zgraja | |

## **Temat:** Dokumentacja projektu

**Ocena:**

.....

**Punkty:**

.....

**Data:**

.....

**Spis treści**

# 1. Założenia i opis funkcjonalny programu

## 1.1. Wstępne informacje

„Middle Ages” jest prostą grą strategiczną, w której Gracz rozbudowuje zamek ulepszając budynki kosztem zebranych zasobów, rekrutuje i szkoli wojowników, a ostatecznie najeżdża wrogów królestwa i zdobywa skarby.

## 1.2. Budynki

Ulepszenie budynku jest możliwe gdy Gracz posiada określoną ilość zasobów. Po kliknięciu przycisku „Upgrade” zasoby zostają zabrane ze skarbca, a statystyki budynku ulegają polepszeniu – przyrost konkretnych surowców na sekundę zwiększa się. Ulepszenie *tartaku* (sawmill) powoduje zwiększenie przyrostu drewna (wood), *farmy* (farm) jedzenia (food), *kamieniołomu* (stone pit) kamienia (stone), *kopalni żelaza* (iron mine) żelaza (iron), *chaty* (hut) wpływów złota z podatków oraz limitu mieszkańców (im większy, tym liczniejszą armię można zwerbować), a *koszarów* (barracks) odblokowuje możliwość zakupu kolejno: *pikiniera* (pikeman), *strzelca* (archer) oraz *jeźdźca* (horseman).

## 1.3. Zarządzanie armią i systemem najazdów na wroga

Żołnierzy można zakupić pod warunkiem posiadania wymaganej liczby zasobów oraz nieprzekroczenia limitu jednostek. Poszczególne jednostki charakteryzują się różną siłą bojową – najsłabszy jest pikinier, a najsilniejszy i zarazem odpowiednio droższy jeździec. Rekrutując oraz ulepszając żołnierzy, siła armii się zwiększa. Jeśli całkowita siła jednostek Gracza jest większa od siły przeciwnika, atak na wrogą siedzibę się powiedzie, jednak losowa ilość żołnierzy Gracza może polec na polu bitwy. W przeciwnym wypadku wszystkie jednostki Gracza zginą. Jeśli najazd się uda, Gracz zdobywa nagrodę w postaci określonej liczby zasobów, a miejsce wrogiego dowódcy zajmuje nowy, silniejszy.

### 1.4. Pobór jedzenia przez żołnierzy i dezercja

Armia musi jeść. W pewnych odstępach czasowych, w zależności od ilości żołnierzy, z konta Gracza zostanie zabrana określona ilość jedzenia. Jeśli Gracz nie będzie w stanie wykarmić swoich wojaków, ich morale znacznie spadną i rozpoczną dezercję. Liczba uciekających żołnierzy jest losowa.

### 1.5. Zbyt mała ilość surowców, limit armii

Jeśli Gracz chce kupić / ulepszyć określony budynek / jednostkę, a nie posiada w skarbcu wymaganych zasobów, bądź zakup jednostki spowodowałby przekroczenie limitu armii, wyświetlony zostanie stosowny komunikat i operacja nie powiedzie się.

### 1.6. Zbiory elementów występujących w grze

*a) Spis budynków:*

- Sawmill (Tartak)
- Farm (Farma)
- Stone pit (Kamieniołom)
- Iron Mine (Kopalnia żelaza)
- Hut (Chata)
- Barracks (Koszary)

*b) Spis zasobów:*

- Gold (Złoto)
- Wood (Drewno)
- Stone (Kamień)
- Iron (Żelazo)
- Food (Jedzenie)

*c) Spis jednostek:*

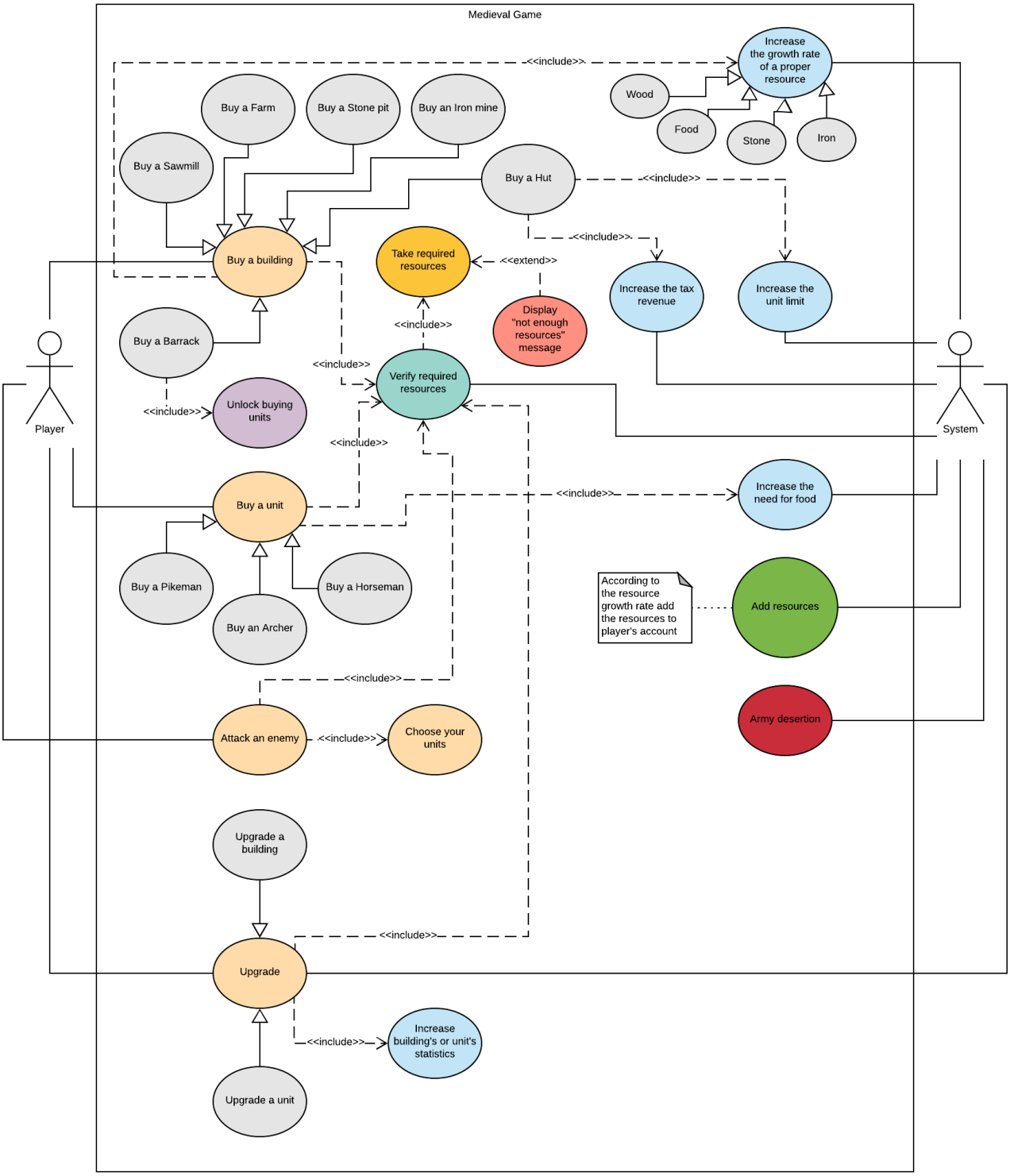
- Pikeman (Pikinier)
- Archer (Strzelec)
- Horseman (Jeździec)

*d) Spis wrogów:*

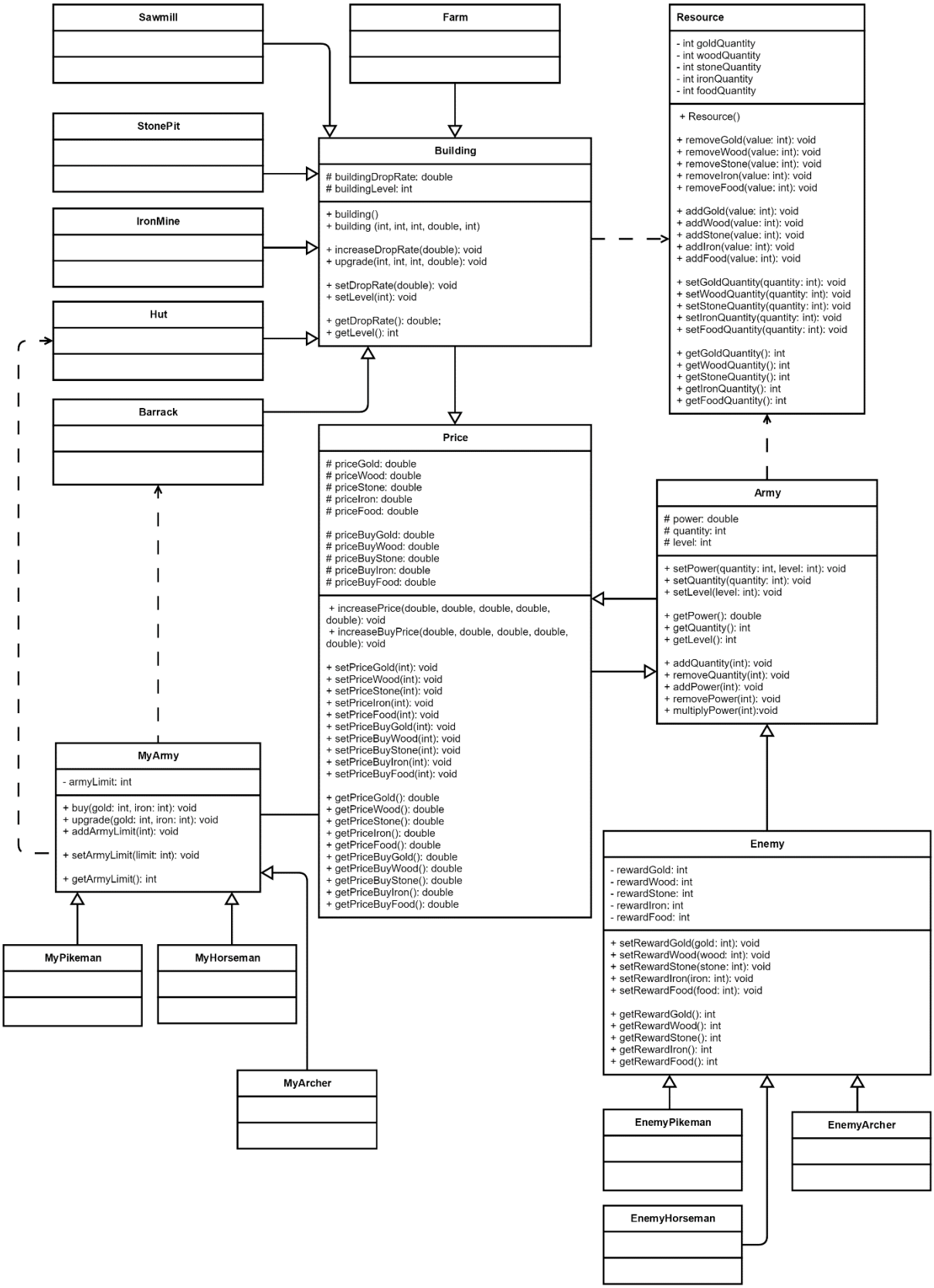
- *Mieszko (poziom - 1, siła – 10)*
- *Gandalf (poziom – 2, siła – 30)*
- *Geralt (poziom – 3, siła – 60)*
- *Helga (poziom – 4, siła – 90)*

# 2. Diagramy UML

## 2.1. Diagram przypadków użycia



## 2.2. Diagram klas



# 3. Spis klas wraz z opisem

## 3.1. Price.cs

**Opis:** Klasa Price.cs odpowiada za zarządzanie cenami.

**Dziedziczenie:** ---

**Metody:** IncreaseUpgradePrice – podnosi cenę ulepszenia

IncreaseBuyPrice – podnosi cenę zakupu

Gettery i settery [1]

**Atrybuty:** priceBuy(*)[2] – przechowuje cenę zakupu

priceUpgrade(*) – przechowuje cenę ulepszenia

## 3.2. Resource.cs

**Opis:** Klasa Resource.cs odpowiada za zarządzanie zasobami.

**Dziedziczenie:** ---

**Metody:** konstruktor

Remove(*) – usunięcie zasobu ze skarbca Gracza

Add(*) – dodanie zasobów do skarbca Gracza

Gettery i Settery

**Atrybuty:** goldQuantity – przechowuje ilość złota

woodQuantity – przechowuje ilość drewna

stoneQuantity – przechowuje ilość kamienia

ironQuantity – przechowuje ilość żelaza

foodQuantity – przechowuje ilość jedzenia

[1] Odpowiadają za ustawianie i pobieranie danej wartości

[2] (*) – w to miejsce wpisać Gold, Wood, Stone, Iron, Food. Osobna funkcja / atrybut występuje dla każdego z tych zasobów

## 3.3. Building.cs

**Opis:** Klasa Building.cs odpowiada za zarządzanie budynkami.

**Dziedziczenie:** Price.cs

**Metody:** IncreaseDropRate – zwiększenie pozyskiwania zasobów

Upgrade – ulepszenie budynku

Gettery i Settery

**Atrybuty:** buildingDropRate – przechowuje tempo pozyskiwania zasobów

buildingLevel – przechowuje poziom danego budynku

### *3.3.1. Sawmill.cs*

**Dziedziczenie:** Building.cs

**Metody:** Konstruktor

**Atrybuty:** ---

### *3.3.2. Farm.cs*

**Dziedziczenie:** Building.cs

**Metody:** Konstruktor

**Atrybuty:** ---

### *3.3.3. StonePit.cs*

**Dziedziczenie:** Building.cs

**Metody:** Konstruktor

**Atrybuty:** ---

### *3.3.4. IronMine.cs*

**Dziedziczenie:** Building.cs

**Metody:** Konstruktor

**Atrybuty:** ---

#### 3.3.5. *Hut.cs*

**Dziedziczenie:** Building.cs

**Metody:** Konstruktor

**Atrybuty:** ---

#### 3.3.6. *Barracks.cs*

**Dziedziczenie:** Building.cs

**Metody:** Konstruktor

**Atrybuty:** ---

## 3.4. Army.cs

**Opis:** Klasa Army.cs odpowiada za zarządzanie armią.

**Dziedziczenie:** Price.cs

**Metody:** AddQuantity – zwiększenie ilości danej jednostki

RemoveQuantity – zmniejszenie ilości danej jednostki

AddPower – zwiększenie siły bojowej

RemovePower – zmniejszenie siły bojowej

MultiplyPower – mnożenie siły bojowej

Gettery i Settery

**Atrybuty:** power – przechowuje siłę bojową

quantity – przechowuje liczebność danej jednostki

level – przechowuje poziom danej jednostki

#### 3.4.1. *Enemy.cs*

**Opis:** Klasa Enemy.cs dotyczy przeciwnika.

**Dziedziczenie:** Army.cs

**Metody:** konstruktor

Gettery i Settery

**Atrybuty:** reward(*) – przechowuje ilość danego zasobu, którą gracz otrzyma po pokonaniu przeciwnika

### 3.4.2. *MyArmy.cs*

**Opis:** Klasa MyArmy.cs odpowiada za zarządzanie armią Gracza.

**Dziedziczenie:** Army.cs

**Metody:** konstruktor

AddArmyLimit – zwiększenie limitu armii

Buy – zakup danej jednostki

Upgrade – ulepszenie danej jednostki

Gettery i Settery

**Atrybuty:** armyLimit – przechowuje limit armii

#### 3.4.2.1. Pikeman.cs

**Dziedziczenie:** MyArmy.cs

**Metody:** Konstruktor

**Atrybuty:** ---

#### 3.4.2.2. Archer.cs

**Dziedziczenie:** MyArmy.cs

**Metody:** Konstruktor

**Atrybuty:** ---

#### 3.4.2.3. Horseman.cs

**Dziedziczenie:** MyArmy.cs

**Metody:** Konstruktor

**Atrybuty:** ---

# 4. Schematy blokowe oraz kod własnych funkcji

## 4.1. IncreaseUpgradePrice

Lokalizacja: Price.cs

Schemat blokowy:

Kod (C#):

```
public void IncreaseUpgradePrice(double gold, double wood, double stone, double iron, double food)
{
   _priceUpgradeGold += gold;
   _priceUpgradeWood += wood;
   _priceUpgradeStone += stone;
   _priceUpgradeIron += iron;
   _priceUpgradeFood += food;
}
```

## 4.2. IncreaseBuyPrice

Lokalizacja: Price.cs

Schemat blokowy:

Kod (C#):

```
public void IncreaseBuyPrice(double gold, double wood, double stone, double iron, double food)
{
   _priceBuyGold += gold;
   _priceBuyWood += wood;
   _priceBuyStone += stone;
   _priceBuyIron += iron;
   _priceBuyFood += food;
}
```

## 4.3. IncreaseDropRate

Lokalizacja: Building.cs

Schemat blokowy:

Kod (C#):

```
public void IncreaseDropRate(double _value)
{
   _buildingDropRate *= _value;
}
```

## 4.4. Upgrade

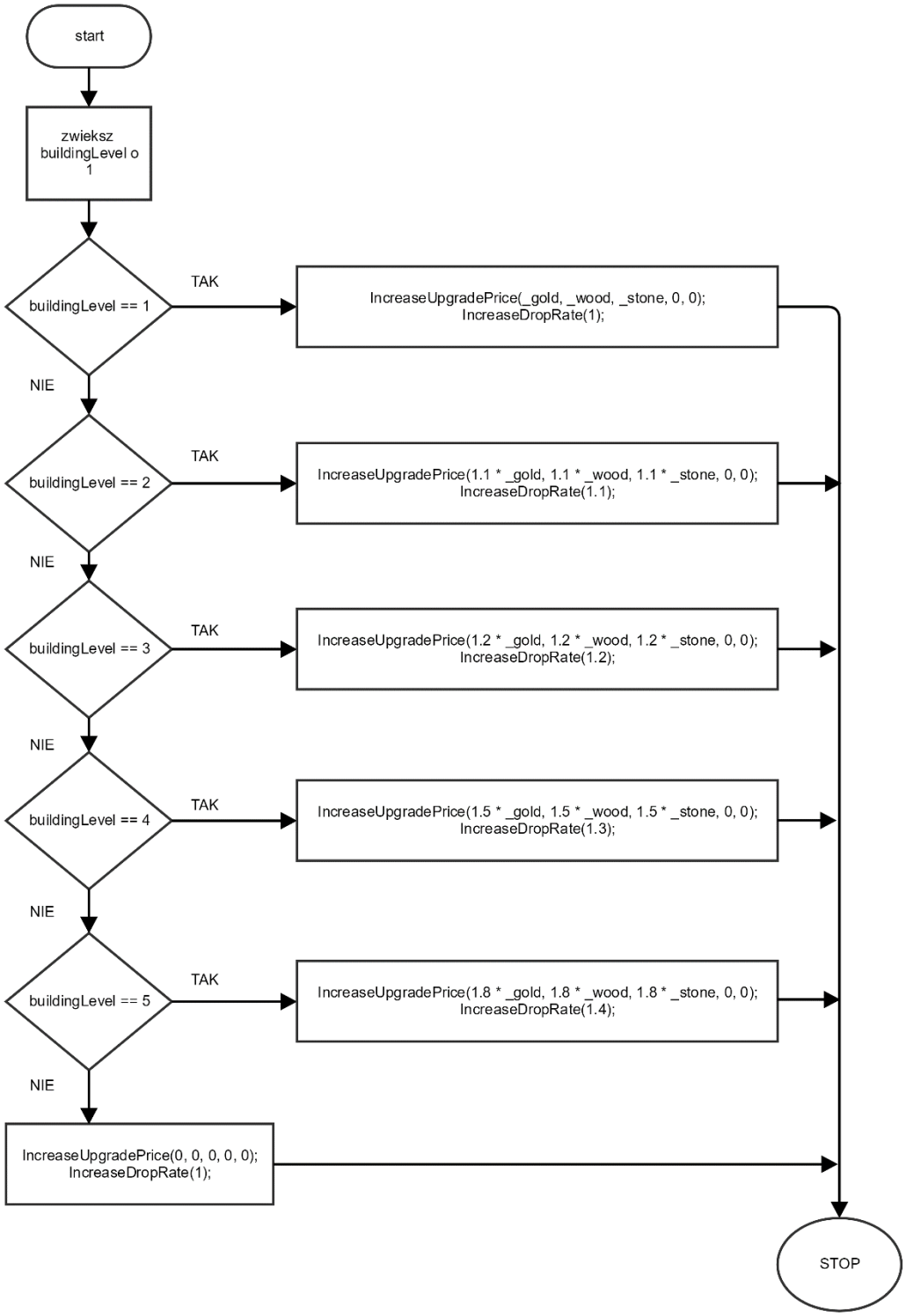
Lokalizacja: Building.cs

Schemat blokowy:

Kod (C#):

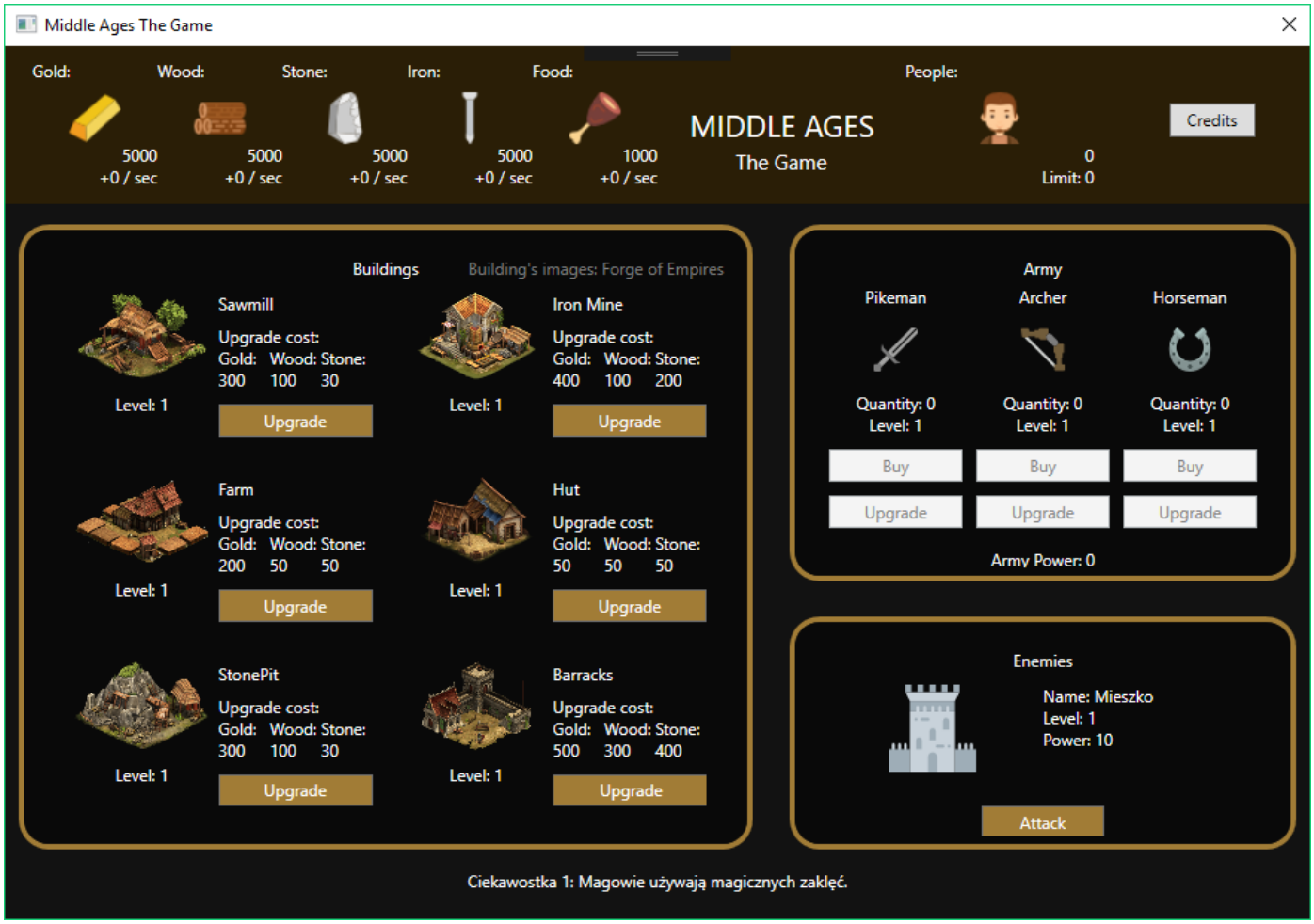
```
public void Upgrade(int _gold, int _wood, int _stone)
{
   _buildingLevel += 1;
   switch (_buildingLevel)
   {
      case 1:
         IncreaseUpgradePrice(_gold, _wood, _stone, 0, 0);
         IncreaseDropRate(1);
         break;
      case 2:
         IncreaseUpgradePrice(1.1 * _gold, 1.1 * _wood, 1.1 * _stone, 0, 0);
         IncreaseDropRate(1.1);
         break;
      case 3:
         IncreaseUpgradePrice(1.2 * _gold, 1.2 * _wood, 1.2 * _stone, 0, 0);
         IncreaseDropRate(1.2);
         break;
      case 4:
         IncreaseUpgradePrice(1.5 * _gold, 1.5 * _wood, 1.5 * _stone, 0, 0);
         IncreaseDropRate(1.3);
         break;
      case 5:
         IncreaseUpgradePrice(1.8 * _gold, 1.8 * _wood, 1.8 * _stone, 0, 0);
         IncreaseDropRate(1.4);
         break;
      default:
         IncreaseUpgradePrice(0, 0, 0, 0, 0);
         IncreaseDropRate(1);
         break;
   }
}
```

Schemat blokowy:



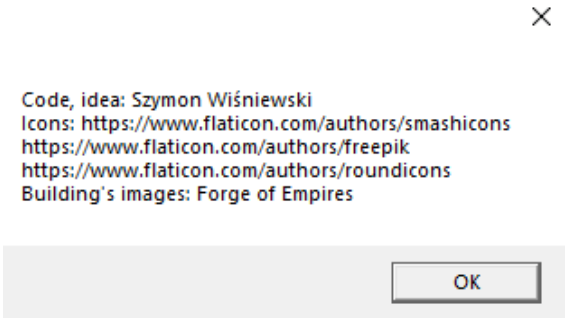
# 5. Opis użytkowy programu – prezentacja

## 5.1. Główne okno




*Screenshot 1: główne okno programu*

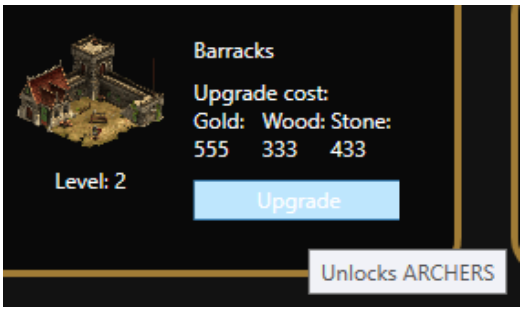
W górnej części okna, po lewej stronie, widnieje aktualna ilość posiadanych przez Gracza zasobów oraz ich przyrost na sekundę. Na tej samej wysokości, po prawej, można sprawdzić liczebność armii oraz maksymalną ilość wojaków, których Gracz może zrekrutować.



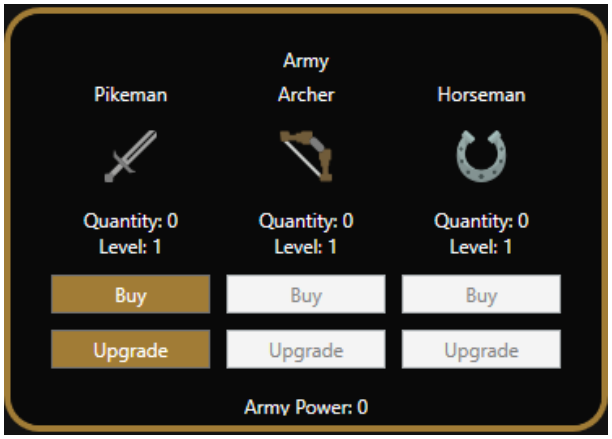

*Screenshot 2: okienko "Credits"*

Przycisk „Credits” otwiera okienko (*screenshot 2.*) z informacjami dotyczącymi autorów ikonek oraz źródłem obrazków użytych do zilustrowania budynków.

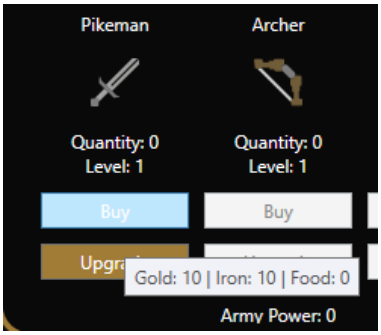
W bloku „Buildings” widoczne są wszystkie dostępne budynki wraz z ikoną, nazwą, kosztami ulepszenia, aktualnym poziomem oraz przyciskiem „Upgrade” umożliwiającym ulepszenie danego obiektu. W przypadku przycisku związanego z Koszarami (Barracks) wyświetlany jest tooltip (*screenshot 3.*) informujący o typie jednostki, która zostanie odblokowana po ulepszeniu budynku. Wtedy przyciski obsługujące przyciski związane z ową jednostką zostają odblokowane (*screenshot 4.*) i Gracz po spełnieniu wymagań (posiadanie odpowiedniej ilości zasobów i nieprzekroczenie limitu podwładnych) ma prawo z nich korzystać. W panelu „Army” gracz może także sprawdzić ilość aktualnie posiadanych żołnierzy oraz ich poziom.




*Screenshot 3: podpowiedzi na przycisku*




*Screenshot 4: panel „Army”*



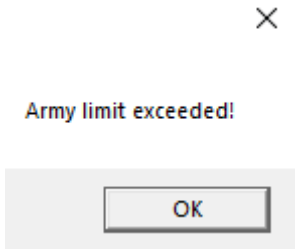

*Screenshot 5: tooltipy w panelu "Army"*

Przyciski „Buy” i „Upgrade” dla każdej jednostki, podobnie jak przycisk „Upgrade” Koszar, mają swoje tooltipy (*screenshot 5.*). Ukazują one koszty wykonania danej operacji – zakupu jednostki, bądź jej ulepszenia. W linijce „Army power” gracz dowiaduje się o aktualnej sile bojowej swej armii.

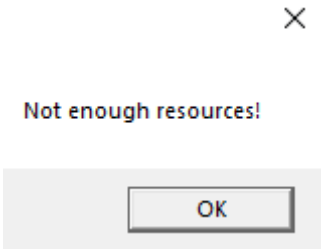

*Screenshot 6: panel "Enemies"*

W panelu „Enemies” widocznym na screenshot 6. Gracz uzyskuje informacje dotyczące przeciwkika – jego nazwę, poziom trudności oraz siłę bojową.

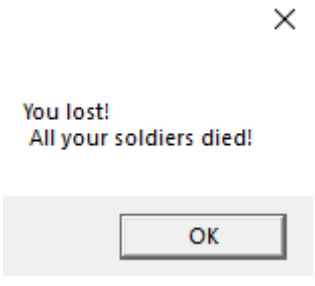
## 5.2. Pozostałe okienka:



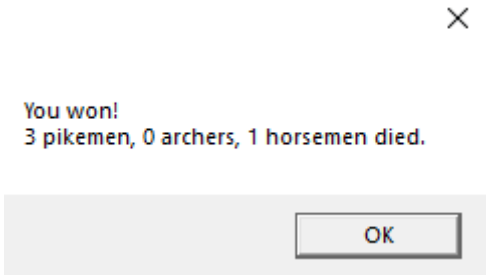

*Screenshot 7: limit armii przekroczony*



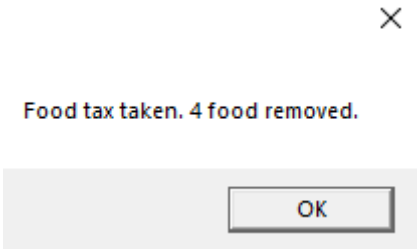

*Screenshot 8: zbyt mało zasobów*



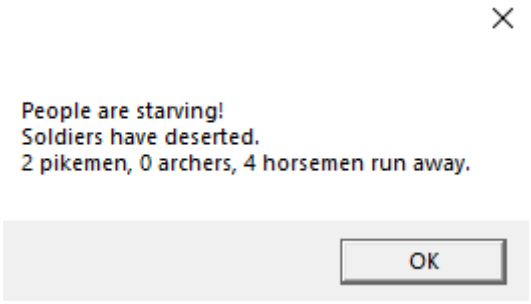

*Screenshot 9: atak na wroga zakończony porażką*




*Screenshot 10: zwycięstwo, informacje o stratach na polu bitwy*




*Screenshot 11: pobór żywności przez dzielnych wojaków*




*Screenshot 12: informacja o dezercji armii*

# 6. Listing kodu C++ / C#

## 6.1. Listing kodu C++

### *6.1.1. mainwindow.cpp*

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>
#include <QTimer>
#include <QDateTime>
#include <string>
#include "time.h"

#include "sawmill.h"
#include "stonepit.h"
#include "ironmine.h"
#include "farm.h"
#include "hut.h"
#include "barracks.h"

#include "resource.h"
#include "llimitexceedederror.h"
#include "noresourceerror.h"
#include "credits.h"
#include "youdied.h"
#include "fooddelete.h"
#include "desertiondialog.h"

#include "pikeman.h"
#include "horseman.h"
#include "archer.h"

#include "enemy.h"

//#####################################
// Objects declaration
SawMill tartak(150, 50, 30, 1.0, 1);
StonePit kamieniolom(300, 100, 30, 1.0, 1);
Farm farma(200, 50, 50, 1.0, 1);
IronMine kopalnia_zelaza(400, 100, 200, 1.0, 1);
Hut domek(50, 50, 50, 1.0, 1);
Barracks koszary(500, 300, 400, 1.0, 1);

MyArmy armia(0); // army limit
Pikeman piknier(0, 0, 1, 10, 10, 10); // power, quantity, level, gold, food,
iron
Archer strzelec(0, 0, 1, 20, 20, 20);
Horseman jezdziec(0, 0, 1, 30, 30, 30);

Enemy wrog(10, 500, 500, 500, 500, 500);
```

```
Resource zasoby(5000, 5000, 5000, 5000, 1000);
//#####################################

//*************************************************************************
*******
//*************************************************************************
*******
//*************************************************************************
*******

//#####################################
// POLYMORPHISM, why didn't I learn that while writing the project :(
class Napis
{
public:
    virtual std::string pobierzTekst() = 0;
};

class Ciekawostka1 :public Napis
{
    std::string tekst;
public:
    Ciekawostka1(std::string x)
    {
        tekst = x;
    }

    std::string pobierzTekst()
    {
        return tekst;
    }
};

class Ciekawostka2 :public Napis
{
    std::string tekst;
public:
    Ciekawostka2(std::string x)
    {
        tekst = x;
    }

    std::string pobierzTekst()
    {
        return tekst;
    }
};
```

```
//#####################################

//*************************************************************************
*******
//*************************************************************************
*******
//*************************************************************************
*******

//#####################################
// CONSTRUCTOR
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    // initialize randomize timer
    //srand(time(NULL));
    qsrand(QDateTime::currentMSecsSinceEpoch() / 1000);

    //#####################################
    // set fixed window size
    this->setFixedSize(QSize(970, 654));
    //#####################################

    //#####################################
    // SLOTS
    connect(ui->pushButton_sawmillUpgrade,
SIGNAL(released()),this,SLOT(upgradeBuildingButtonPressed()));
    connect(ui->pushButton_farmUpgrade,
SIGNAL(released()),this,SLOT(upgradeBuildingButtonPressed()));
    connect(ui->pushButton_hutUpgrade,
SIGNAL(released()),this,SLOT(upgradeBuildingButtonPressed()));
    connect(ui->pushButton_ironMineUpgrade,
SIGNAL(released()),this,SLOT(upgradeBuildingButtonPressed()));
    connect(ui->pushButton_stonePitUpgrade,
SIGNAL(released()),this,SLOT(upgradeBuildingButtonPressed()));
    connect(ui->pushButton_barracksUpgrade,
SIGNAL(released()),this,SLOT(upgradeBuildingButtonPressed()));

    connect(ui->pushButton_Upgrade_Archer,
SIGNAL(released()),this,SLOT(upgradeArmyButtonPressed()));
    connect(ui->pushButton_Upgrade_Horseman,
SIGNAL(released()),this,SLOT(upgradeArmyButtonPressed()));
    connect(ui->pushButton_Upgrade_Pikeman,
SIGNAL(released()),this,SLOT(upgradeArmyButtonPressed()));
```

```
    connect(ui->pushButton_Buy_Archer,
SIGNAL(released()),this,SLOT(buyArmyButtonPressed()));
    connect(ui->pushButton_Buy_Horseman,
SIGNAL(released()),this,SLOT(buyArmyButtonPressed()));
    connect(ui->pushButton_Buy_Pikeman,
SIGNAL(released()),this,SLOT(buyArmyButtonPressed()));

    connect(ui->pushButtonCredits,
SIGNAL(released()),this,SLOT(creditsButtonPressed()));

    connect(ui->pushButtonAttackEnemy,
SIGNAL(released()),this,SLOT(attackEnemyButton()));
    //#####################################

    //#####################################
    // TIMERS
    timerGold = new QTimer(this);
    connect(timerGold, SIGNAL(timeout()), this, SLOT(goldIncome()));
    timerGold->start(1000);

    timerWood = new QTimer(this);
    connect(timerWood, SIGNAL(timeout()), this, SLOT(woodIncome()));
    timerWood->start(1000);

    timerStone = new QTimer(this);
    connect(timerStone, SIGNAL(timeout()), this, SLOT(stoneIncome()));
    timerStone->start(1000);

    timerIron = new QTimer(this);
    connect(timerIron, SIGNAL(timeout()), this, SLOT(ironIncome()));
    timerIron->start(1000);

    timerFood = new QTimer(this);
    connect(timerFood, SIGNAL(timeout()), this, SLOT(foodIncome()));
    timerFood->start(1000);

    timerFoodMinus = new QTimer(this);
    connect(timerFoodMinus, SIGNAL(timeout()), this, SLOT(foodMinus()));
    timerFoodMinus->start(10000); // 1 minute
    //#####################################

    //#####################################
    // DEFINE STARTING DROP VALUES
    goldDropValue = 0;
    woodDropValue = 0;
    stoneDropValue = 0;
    ironDropValue = 0;
    foodDropValue = 0;
    //#####################################
```

```
    //#####################################
    // SET DEFAULT LABELS
    // sawmill
    ui->label_sawmillLevel->setText("Level: " +
QString::number(tartak.getLevel(), 'g', 15));
    ui->label_sawmillPrice_2->setText("G: " +
QString::number(tartak.getPriceGold(), 'g', 15) + " | W: " +
QString::number(tartak.getPriceWood(), 'g', 15) + " | S: " +
QString::number(tartak.getPriceStone(), 'g', 15));
    ui->label_woodIncome->setText("+" + QString::number(woodDropValue) + " /
sec");

    // stone pit
    ui->label_stonePitLevel->setText("Level: " +
QString::number(kamieniolom.getLevel(), 'g', 15));
    ui->label_stonePitPrice_2->setText("G: " +
QString::number(kamieniolom.getPriceGold(), 'g', 15) + " | W: " +
QString::number(kamieniolom.getPriceWood(), 'g', 15) + " | S: " +
QString::number(kamieniolom.getPriceStone(), 'g', 15));
    ui->label_stoneIncome->setText("+" + QString::number(stoneDropValue) + " /
sec");

    // iron mine
    ui->label_ironMineLevel->setText("Level: " +
QString::number(kopalnia_zelaza.getLevel(), 'g', 15));
    ui->label_ironMinePrice_2->setText("G: " +
QString::number(kopalnia_zelaza.getPriceGold(), 'g', 15) + " | W: " +
QString::number(kopalnia_zelaza.getPriceWood(), 'g', 15) + " | S: " +
QString::number(kopalnia_zelaza.getPriceStone(), 'g', 15));
    ui->label_ironIncome->setText("+" + QString::number(ironDropValue) + " /
sec");

    // farm
    ui->label_farmLevel->setText("Level: " + QString::number(farma.getLevel(),
'g', 15));
    ui->label_farmPrice_2->setText("G: " +
QString::number(farma.getPriceGold(), 'g', 15) + " | W: " +
QString::number(farma.getPriceWood(), 'g', 15) + " | S: " +
QString::number(farma.getPriceStone(), 'g', 15));
    ui->label_foodIncome->setText("+" + QString::number(foodDropValue) + " /
sec");

    // hut
    ui->label_hutLevel->setText("Level: " + QString::number(domek.getLevel(),
'g', 15));
    ui->label_hutPrice_2->setText("G: " +
QString::number(domek.getPriceGold(), 'g', 15) + " | W: " +
```

```
QString::number(domek.getPriceWood(), 'g', 15) + " | S: " +
QString::number(domek.getPriceStone(), 'g', 15));
    ui->label_goldIncome->setText("+" + QString::number(goldDropValue) + " /
sec");
    // PEOPLE LIMIT

    // barracks
    ui->label_barracksLevel->setText("Level: " +
QString::number(koszary.getLevel(), 'g', 15));
    ui->label_barracksPrice_2->setText("G: " +
QString::number(koszary.getPriceGold(), 'g', 15) + " | W: " +
QString::number(koszary.getPriceWood(), 'g', 15) + " | S: " +
QString::number(koszary.getPriceStone(), 'g', 15));
    // ARMY LIMIT ui->label_foodIncome->setText("+" +
QString::number(foodDropValue) + " / sec");

    // resource values
    ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
    ui->label_woodValue->setText(QString::number(zasoby.getWoodQuantity(),
'g', 15));
    ui->label_stoneValue->setText(QString::number(zasoby.getStoneQuantity(),
'g', 15));
    ui->label_ironValue->setText(QString::number(zasoby.getIronQuantity(),
'g', 15));
    ui->label_foodValue->setText(QString::number(zasoby.getFoodQuantity(),
'g', 15));

    // army
    ui->labelPikemanQuantity->setText("Quantity: " +
QString::number(piknier.getQuantity(), 'g', 15));
    ui->labelPikemanLevel->setText("Level: " +
QString::number(piknier.getLevel(), 'g', 15));
    ui->labelArcherQuantity->setText("Quantity: " +
QString::number(strzelec.getQuantity(), 'g', 15));
    ui->label_ArcherLevel->setText("Level: " +
QString::number(strzelec.getLevel(), 'g', 15));
    ui->labelHorsemanQuantity->setText("Quantity: " +
QString::number(jezdziec.getQuantity(), 'g', 15));
    ui->label_HorsemanLevel->setText("Level: " +
QString::number(jezdziec.getLevel(), 'g', 15));

    ui->labelPikemanQuantity->setText("Quantity: " +
QString::number(piknier.getQuantity()));
    ui->label_peopleValue->setText("People: " +
QString::number(piknier.getQuantity() + strzelec.getQuantity() +
jezdziec.getQuantity()));

    // people limit
```

```
    ui->label_peopleLimit->setText(" / " +
QString::number(armia.getArmyLimit(), 'g', 15));

    // update army power label
    ui->labelArmyPower->setText("Army power: " +
QString::number(piknier.getPower() + jezdziec.getPower() +
strzelec.getPower(), 'g', 15));

    // update enemy labels
    ui->label_enemyLevel->setText("Level: 1");
    ui->label_enemyName->setText("Name: Mieszko");
    ui->label_enemyPower->setText("Power: " + QString::number(wrog.getPower(),
'g', 15));
    //#####################################

    //#####################################
    // disable some buttons
    ui->pushButton_Buy_Pikeman->setEnabled(false);
    ui->pushButton_Buy_Archer->setEnabled(false);
    ui->pushButton_Buy_Horseman->setEnabled(false);
    ui->pushButton_Upgrade_Horseman->setEnabled(false);
    ui->pushButton_Upgrade_Archer->setEnabled(false);
    ui->pushButton_Upgrade_Pikeman->setEnabled(false);
    //#####################################

    //#####################################
    // army button tooltips
    ui->pushButton_Buy_Pikeman->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                           "<ul>"
                                           "<li>Gold: " +
QString::number(piknier.getPriceBuyGold()) + "</li>"
                                           "<li>Food: " +
QString::number(piknier.getPriceBuyFood()) + "</li>"
                                           "<li>Iron: " +
QString::number(piknier.getPriceBuyIron()) + "</li>"
                                           "</ul>");
    ui->pushButton_Buy_Horseman->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                           "<ul>"
                                           "<li>Gold: " +
QString::number(jezdziec.getPriceBuyGold()) + "</li>"
                                           "<li>Food: " +
QString::number(jezdziec.getPriceBuyFood()) + "</li>"
                                           "<li>Iron: " +
QString::number(jezdziec.getPriceBuyIron()) + "</li>"
                                           "</ul>");
    ui->pushButton_Buy_Archer->setToolTip("<font
color='red'><b>Cost:</b></font> "
```

```
                                            "<ul>"
                                            "<li>Gold: " +
QString::number(strzelec.getPriceBuyGold()) + "</li>"
                                            "<li>Food: " +
QString::number(strzelec.getPriceBuyFood()) + "</li>"
                                            "<li>Iron: " +
QString::number(strzelec.getPriceBuyIron()) + "</li>"
                                            "</ul>");

    ui->pushButton_Upgrade_Pikeman->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                            "<ul>"
                                            "<li>Gold: " +
QString::number(piknier.getPriceGold()) + "</li>"
                                            "<li>Food: " +
QString::number(piknier.getPriceFood()) + "</li>"
                                            "<li>Iron: " +
QString::number(piknier.getPriceIron()) + "</li>"
                                            "</ul>");
    ui->pushButton_Upgrade_Horseman->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                            "<ul>"
                                            "<li>Gold: " +
QString::number(jezdziec.getPriceGold()) + "</li>"
                                            "<li>Food: " +
QString::number(jezdziec.getPriceFood()) + "</li>"
                                            "<li>Iron: " +
QString::number(jezdziec.getPriceIron()) + "</li>"
                                            "</ul>");
    ui->pushButton_Upgrade_Archer->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                            "<ul>"
                                            "<li>Gold: " +
QString::number(strzelec.getPriceGold()) + "</li>"
                                            "<li>Food: " +
QString::number(strzelec.getPriceFood()) + "</li>"
                                            "<li>Iron: " +
QString::number(strzelec.getPriceIron()) + "</li>"
                                            "</ul>");

    // barracks level 1 tooltip
    ui->pushButton_barracksUpgrade->setToolTip("Unlocks <font
color='green'><b>PIKEMEN</b></font>.");

    // building upgrade tooltips
    ui->pushButton_sawmillUpgrade->setToolTip("Increases <font
color='green'><b>WOOD</b></font> drop rate.");
    ui->pushButton_stonePitUpgrade->setToolTip("Increases <font
color='green'><b>STONE</b></font> drop rate.");
```

```
    ui->pushButton_ironMineUpgrade->setToolTip("Increases <font
color='green'><b>IRON</b></font> drop rate.");
    ui->pushButton_farmUpgrade->setToolTip("Increases <font
color='green'><b>FOOD</b></font> drop rate.");
    ui->pushButton_hutUpgrade->setToolTip("Increases <font
color='green'><b>GOLD</b></font> drop rate and <font
color='green'><b>PEOPLE</b></font> limit</font>.");
    //#####################################


    //#####################################
    // POLYMORPHISM, changing the ciekawostka label
    Ciekawostka1 c1("Ciekawostka 1: Magowie uzywaja magicznych zaklec.");
    Ciekawostka2 c2("Ciekawostka2: Wojownicy sa silni.");
    Napis *wsk;

    wsk = &c1;
    ui->Ciekawostka->setText(QString::fromStdString(wsk->pobierzTekst()));
    //#####################################
}

//#####################################

MainWindow::~MainWindow()
{
    delete ui;
}

//*************************************************************************
*******
//*************************************************************************
*******
//*************************************************************************
*******

//#####################################
// FUNCTION THAT CONTROLLS CREDITS BUTTON
void MainWindow::creditsButtonPressed()
{
    Credits creditsDialog;
    creditsDialog.setModal(true);
    creditsDialog.exec();
}
//#####################################

//*************************************************************************
*******
//*************************************************************************
*******
```

```
//*************************************************************************
*******

//#####################################
// RESOURCE INCOME FUNCTIONS
void MainWindow::goldIncome()
{
    zasoby.addGold(goldDropValue);
    ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
}

void MainWindow::woodIncome()
{
    zasoby.addWood(woodDropValue);
    ui->label_woodValue->setText(QString::number(zasoby.getWoodQuantity(),
'g', 15));
}

void MainWindow::stoneIncome()
{
    zasoby.addStone(stoneDropValue);
    ui->label_stoneValue->setText(QString::number(zasoby.getStoneQuantity(),
'g', 15));
}

void MainWindow::ironIncome()
{
    zasoby.addIron(ironDropValue);
    ui->label_ironValue->setText(QString::number(zasoby.getIronQuantity(),
'g', 15));
}

void MainWindow::foodIncome()
{
    zasoby.addFood(foodDropValue);
    ui->label_foodValue->setText(QString::number(zasoby.getFoodQuantity(),
'g', 15));
}

void MainWindow::foodMinus()
{
    int x = piknier.getQuantity() + jezdziec.getQuantity() +
strzelec.getQuantity();

    if(x > 0)
    {
        if(zasoby.getFoodQuantity() - x >= 0)
        {
```

```
            for(int i = 0; i < x; i++)
                zasoby.removeFood(1);

            ui->label_foodValue-
>setText(QString::number(zasoby.getFoodQuantity(), 'g', 15));

            // open up the dialog window
            foodDelete deleteFood;
            deleteFood.setModal(true);
            deleteFood.exec();
        }

        // if food quantity < 0 after the food tax units start deserting
        else {
            desertion();
        }
    }
}

void MainWindow::desertion()
{
    if(piknier.getQuantity() > 0)
    {
        int randomNum = static_cast<int>(qrand() % piknier.getQuantity());
        piknier.removeQuantity(randomNum);
        piknier.removePower(randomNum * 5);
    }

    if(strzelec.getQuantity() > 0)
    {
        int randomNum2 = static_cast<int>(qrand() % strzelec.getQuantity());
        strzelec.removeQuantity(randomNum2);
        strzelec.removePower(randomNum2 * 7);
    }

    if(jezdziec.getQuantity() > 0)
    {
        int randomNum3 = static_cast<int>(qrand() % jezdziec.getQuantity());
        jezdziec.removeQuantity(randomNum3);
        jezdziec.removePower(randomNum3 * 10);
    }

    //update power label
    ui->labelArmyPower->setText("Army power: " +
QString::number(piknier.getPower() + strzelec.getPower() +
jezdziec.getPower(), 'g', 15));

    // update unit quantity labels
```

```
    ui->labelPikemanQuantity->setText("Quantity: " +
QString::number(piknier.getQuantity(), 'g', 15));
    ui->labelArcherQuantity->setText("Quantity: " +
QString::number(strzelec.getQuantity(), 'g', 15));
    ui->labelHorsemanQuantity->setText("Quantity: " +
QString::number(jezdziec.getQuantity(), 'g', 15));
    ui->label_peopleValue->setText("People: " +
QString::number(piknier.getQuantity() + strzelec.getQuantity() +
jezdziec.getQuantity(), 'g', 15));

    desertionDialog desertionmessage;
    desertionmessage.setModal(true);
    desertionmessage.exec();
}
//#####################################

//*************************************************************************
*******
//*************************************************************************
*******
//*************************************************************************
*******

//#####################################
// Function controlling "upgrade building" button
void MainWindow::upgradeBuildingButtonPressed()
{
    QPushButton * button = (QPushButton*)sender();


    // SawMill
    if ((button == ui->pushButton_sawmillUpgrade) && (tartak.getLevel() < 6)
&& (zasoby.getGoldQuantity() - tartak.getPriceGold() >= 0) &&
(zasoby.getWoodQuantity() - tartak.getPriceWood() >= 0) &&
(zasoby.getStoneQuantity() - tartak.getPriceStone() >= 0))
    {
        woodDropValue++;

        // remove resources and update resource values
        zasoby.removeGold(tartak.getPriceGold());
        zasoby.removeWood(tartak.getPriceWood());
        zasoby.removeStone(tartak.getPriceStone());

        tartak.upgrade(50, 30, 30, 1.0);

        // update level label and resource value labels
        ui->label_sawmillLevel->setText("Level: " +
QString::number(tartak.getLevel(), 'g', 15));
```

```
        ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
        ui->label_woodValue->setText(QString::number(zasoby.getWoodQuantity(),
'g', 15));
        ui->label_stoneValue-
>setText(QString::number(zasoby.getStoneQuantity(), 'g', 15));

        // update income label
        ui->label_woodIncome->setText("+" + QString::number(woodDropValue) + "
/ sec");

        // update price labels
        if(ui->label_sawmillLevel->text() == "Level: 6")
            ui->label_sawmillPrice_2->setText("Max level reached");
        else
            ui->label_sawmillPrice_2->setText("G: " +
QString::number(tartak.getPriceGold(), 'g', 15) + " | W: " +
QString::number(tartak.getPriceWood(), 'g', 15) + " | S: " +
QString::number(tartak.getPriceStone(), 'g', 15));
    }

    // Stone Pit
    else if ((button == ui->pushButton_stonePitUpgrade) &&
(kamieniolom.getLevel() < 6) && (zasoby.getGoldQuantity() -
kamieniolom.getPriceGold() >= 0) && (zasoby.getWoodQuantity() -
kamieniolom.getPriceWood() >= 0) && (zasoby.getStoneQuantity() -
kamieniolom.getPriceStone() >= 0))
    {
        stoneDropValue++;

        // remove resources and update resource values
        zasoby.removeGold(kamieniolom.getPriceGold());
        zasoby.removeWood(kamieniolom.getPriceWood());
        zasoby.removeStone(kamieniolom.getPriceStone());

        kamieniolom.upgrade(50, 30, 30, 1.0);

        // update level label and resource value labels
        ui->label_stonePitLevel->setText("Level: " +
QString::number(kamieniolom.getLevel(), 'g', 15));
        ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
        ui->label_woodValue->setText(QString::number(zasoby.getWoodQuantity(),
'g', 15));
        ui->label_stoneValue-
>setText(QString::number(zasoby.getStoneQuantity(), 'g', 15));

        // update income label
```

```
        ui->label_stoneIncome->setText("+" + QString::number(stoneDropValue) +
" / sec");

        // update price labels
        if(ui->label_stonePitLevel->text() == "Level: 6")
            ui->label_stonePitPrice_2->setText("Max level reached");
        else
            ui->label_stonePitPrice_2->setText("G: " +
QString::number(kamieniolom.getPriceGold(), 'g', 15) + " | W: " +
QString::number(kamieniolom.getPriceWood(), 'g', 15) + " | S: " +
QString::number(kamieniolom.getPriceStone(), 'g', 15));
    }

    // Iron Mine
    else if ((button == ui->pushButton_ironMineUpgrade) &&
(kopalnia_zelaza.getLevel() < 6) && (zasoby.getGoldQuantity() -
kopalnia_zelaza.getPriceGold() >= 0) && (zasoby.getWoodQuantity() -
kopalnia_zelaza.getPriceWood() >= 0) && (zasoby.getStoneQuantity() -
kopalnia_zelaza.getPriceStone() >= 0))
    {
        ironDropValue++;

        // remove resources and update resource values
        zasoby.removeGold(kopalnia_zelaza.getPriceGold());
        zasoby.removeWood(kopalnia_zelaza.getPriceWood());
        zasoby.removeStone(kopalnia_zelaza.getPriceStone());

        kopalnia_zelaza.upgrade(50, 30, 30, 1.0);

        // update level label and resource value labels
        ui->label_ironMineLevel->setText("Level: " +
QString::number(kopalnia_zelaza.getLevel(), 'g', 15));
        ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
        ui->label_woodValue->setText(QString::number(zasoby.getWoodQuantity(),
'g', 15));
        ui->label_stoneValue-
>setText(QString::number(zasoby.getStoneQuantity(), 'g', 15));

        // update income label
        ui->label_ironIncome->setText("+" + QString::number(ironDropValue) + "
/ sec");

        // update price labels
        if(ui->label_ironMineLevel->text() == "Level: 6")
            ui->label_ironMinePrice_2->setText("Max level reached");
        else
            ui->label_ironMinePrice_2->setText("G: " +
QString::number(kopalnia_zelaza.getPriceGold(), 'g', 15) + " | W: " +
```

```
QString::number(kopalnia_zelaza.getPriceWood(), 'g', 15) + " | S: " +
QString::number(kopalnia_zelaza.getPriceStone(), 'g', 15));
    }

    // FARM
    else if ((button == ui->pushButton_farmUpgrade) && (farma.getLevel() < 6)
&& (zasoby.getGoldQuantity() - farma.getPriceGold() >= 0) &&
(zasoby.getWoodQuantity() - farma.getPriceWood() >= 0) &&
(zasoby.getStoneQuantity() - farma.getPriceStone() >= 0))
    {
        foodDropValue++;

        // remove resources and update resource values
        zasoby.removeGold(farma.getPriceGold());
        zasoby.removeWood(farma.getPriceWood());
        zasoby.removeStone(farma.getPriceStone());

        farma.upgrade(50, 30, 30, 1.0);

        // update level label and resource value labels
        ui->label_farmLevel->setText("Level: " +
QString::number(farma.getLevel(), 'g', 15));
        ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
        ui->label_woodValue->setText(QString::number(zasoby.getWoodQuantity(),
'g', 15));
        ui->label_stoneValue-
>setText(QString::number(zasoby.getStoneQuantity(), 'g', 15));

        // update income label
        ui->label_foodIncome->setText("+" + QString::number(foodDropValue) + "
/ sec");

        // update price labels
        if(ui->label_farmLevel->text() == "Level: 6")
            ui->label_farmPrice_2->setText("Max level reached");
        else
            ui->label_farmPrice_2->setText("G: " +
QString::number(farma.getPriceGold(), 'g', 15) + " | W: " +
QString::number(farma.getPriceWood(), 'g', 15) + " | S: " +
QString::number(farma.getPriceStone(), 'g', 15));
    }

    // Hut
    else if ((button == ui->pushButton_hutUpgrade) && (domek.getLevel() < 6)
&& (zasoby.getGoldQuantity() - domek.getPriceGold() >= 0) &&
(zasoby.getWoodQuantity() - domek.getPriceWood() >= 0) &&
(zasoby.getStoneQuantity() - domek.getPriceStone() >= 0))
    {
```

```
        goldDropValue++;

        // remove resources and update resource values
        zasoby.removeGold(domek.getPriceGold());
        zasoby.removeWood(domek.getPriceWood());
        zasoby.removeStone(domek.getPriceStone());

        domek.upgrade(50, 30, 30, 1.0);
        armia.addArmyLimit(10);

        // update people limit label
        ui->label_peopleLimit->setText(" / " +
QString::number(armia.getArmyLimit(), 'g', 15));

        // update level label and resource value labels
        ui->label_hutLevel->setText("Level: " +
QString::number(domek.getLevel(), 'g', 15));
        ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
        ui->label_woodValue->setText(QString::number(zasoby.getWoodQuantity(),
'g', 15));
        ui->label_stoneValue-
>setText(QString::number(zasoby.getStoneQuantity(), 'g', 15));

        // update income label
        ui->label_goldIncome->setText("+" + QString::number(goldDropValue) + "
/ sec");

        // update price labels
        if(ui->label_hutLevel->text() == "Level: 6")
            ui->label_hutPrice_2->setText("Max level reached");
        else
            ui->label_hutPrice_2->setText("G: " +
QString::number(domek.getPriceGold(), 'g', 15) + " | W: " +
QString::number(domek.getPriceWood(), 'g', 15) + " | S: " +
QString::number(domek.getPriceStone(), 'g', 15));
    }

    // Barracks
    else if ((button == ui->pushButton_barracksUpgrade) && (koszary.getLevel()
< 4) && (zasoby.getGoldQuantity() - koszary.getPriceGold() >= 0) &&
(zasoby.getWoodQuantity() - koszary.getPriceWood() >= 0) &&
(zasoby.getStoneQuantity() - koszary.getPriceStone() >= 0))
    {

        // UPDATE TOOLTIPS AND ENABLE ARMY BUTTONS
        if(koszary.getLevel() == 1)
        {
```

```
            ui->pushButton_barracksUpgrade->setToolTip("Unlocks <font
color='green'><b>ARCHERS</b></font>.");
            ui->pushButton_Buy_Pikeman->setEnabled(true);
            ui->pushButton_Upgrade_Pikeman->setEnabled(true);
        }
        else if(koszary.getLevel() == 2)
        {
            ui->pushButton_barracksUpgrade->setToolTip("unlocks <font
color='green'><b>HORSEMEN</b></font>.");
            ui->pushButton_Buy_Archer->setEnabled(true);
            ui->pushButton_Upgrade_Archer->setEnabled(true);
        }
        else if(koszary.getLevel() == 3)
        {
            ui->pushButton_Buy_Horseman->setEnabled(true);
            ui->pushButton_Upgrade_Horseman->setEnabled(true);
        }

        // remove resources and update resource values
        zasoby.removeGold(koszary.getPriceGold());
        zasoby.removeWood(koszary.getPriceWood());
        zasoby.removeStone(koszary.getPriceStone());

        koszary.upgrade(50, 30, 30, 1.0);

        // update level label and resource value labels
        ui->label_barracksLevel->setText("Level: " +
QString::number(koszary.getLevel(), 'g', 15));
        ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
        ui->label_woodValue->setText(QString::number(zasoby.getWoodQuantity(),
'g', 15));
        ui->label_stoneValue-
>setText(QString::number(zasoby.getStoneQuantity(), 'g', 15));


        // update price labels
        if(ui->label_barracksLevel->text() == "Level: 4")
            ui->label_barracksPrice_2->setText("Max level reached");
        else
            ui->label_barracksPrice_2->setText("G: " +
QString::number(koszary.getPriceGold(), 'g', 15) + " | W: " +
QString::number(koszary.getPriceWood(), 'g', 15) + " | S: " +
QString::number(koszary.getPriceStone(), 'g', 15));
    }

    // OPEN ERROR DIALOG
    else
    {
```

```
        NoResourceError noResourceError;
        noResourceError.setModal(true);
        noResourceError.exec();
    }

    // disable upgrade buttons when level = 6
    if (tartak.getLevel() == 6)
    {
        ui->pushButton_sawmillUpgrade->setEnabled(false);
    }
    if (kamieniolom.getLevel() == 6)
    {
        ui->pushButton_stonePitUpgrade->setEnabled(false);
    }
    if (kopalnia_zelaza.getLevel() == 6)
    {
        ui->pushButton_ironMineUpgrade->setEnabled(false);
    }
    if (farma.getLevel() == 6)
    {
        ui->pushButton_farmUpgrade->setEnabled(false);
    }
    if (domek.getLevel() == 6)
    {
        ui->pushButton_hutUpgrade->setEnabled(false);
    }
    if (koszary.getLevel() == 4)
    {
        ui->pushButton_barracksUpgrade->setEnabled(false);
    }
}
//#####################################

//*************************************************************************
*******
//*************************************************************************
*******
//*************************************************************************
*******

//#####################################
// Function controlling "buy building" button
void MainWindow::buyBuildingButtonPressed()
{

}
//#####################################
```

```
//*************************************************************************
*******
//*************************************************************************
*******
//*************************************************************************
*******

//####################################
// Function controlling "buy army" button
void MainWindow::buyArmyButtonPressed()
{
    QPushButton * button = (QPushButton*)sender();

    int quantity = 1;

    // pikeman
    if ((button == ui->pushButton_Buy_Pikeman)
            && ((piknier.getQuantity() + quantity + strzelec.getQuantity() +
jezdziec.getQuantity()) <= armia.getArmyLimit())
            && (zasoby.getGoldQuantity() - piknier.getPriceBuyGold() >= 0) &&
(zasoby.getFoodQuantity() - piknier.getPriceBuyFood() >= 0) &&
(zasoby.getIronQuantity() - piknier.getPriceBuyIron() >= 0))
    {
        piknier.buy(5, quantity, 10, 10, 10);

        // remove resources and update resource values
        zasoby.removeGold(piknier.getPriceBuyGold());
        zasoby.removeIron(piknier.getPriceBuyIron());
        zasoby.removeFood(piknier.getPriceBuyFood());

        // update piknier quantity and people quantity label
        ui->labelPikemanQuantity->setText("Quantity: " +
QString::number(piknier.getQuantity()));
        ui->label_peopleValue->setText("People: " +
QString::number(piknier.getQuantity() + strzelec.getQuantity() +
jezdziec.getQuantity()));

        // update level label and resource value labels
        ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
        ui->label_ironValue->setText(QString::number(zasoby.getIronQuantity(),
'g', 15));
        ui->label_foodValue->setText(QString::number(zasoby.getFoodQuantity(),
'g', 15));

        // update army power label
        ui->labelArmyPower->setText("Army power: " +
QString::number(piknier.getPower() + strzelec.getPower() +
jezdziec.getPower()));
```

```
        // update pikeman price tooltip
        ui->pushButton_Buy_Pikeman->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                               "<ul>"
                                               "<li>Gold: " +
QString::number(piknier.getPriceBuyGold()) + "</li>"
                                               "<li>Food: " +
QString::number(piknier.getPriceBuyFood()) + "</li>"
                                               "<li>Iron: " +
QString::number(piknier.getPriceBuyIron()) + "</li>"
                                               "</ul>");
    }

    // Archer
    else if ((button == ui->pushButton_Buy_Archer)
            && ((piknier.getQuantity() + quantity + strzelec.getQuantity() +
jezdziec.getQuantity()) <= armia.getArmyLimit())
            && (zasoby.getGoldQuantity() - strzelec.getPriceBuyGold() >= 0) &&
(zasoby.getFoodQuantity() - strzelec.getPriceBuyFood() >= 0) &&
(zasoby.getIronQuantity() - strzelec.getPriceBuyIron() >= 0))
    {
        strzelec.buy(7, quantity, 20, 20, 20);

        // remove resources and update resource values
        zasoby.removeGold(strzelec.getPriceBuyGold());
        zasoby.removeIron(strzelec.getPriceBuyIron());
        zasoby.removeFood(strzelec.getPriceBuyFood());

        // update strzelec quantity and people quantity label
        ui->labelArcherQuantity->setText("Quantity: " +
QString::number(strzelec.getQuantity()));
        ui->label_peopleValue->setText("People: " +
QString::number(piknier.getQuantity() + strzelec.getQuantity() +
jezdziec.getQuantity()));

        // update level label and resource value labels
        ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
        ui->label_ironValue->setText(QString::number(zasoby.getIronQuantity(),
'g', 15));
        ui->label_foodValue->setText(QString::number(zasoby.getFoodQuantity(),
'g', 15));

        // update army power label
        ui->labelArmyPower->setText("Army power: " +
QString::number(piknier.getPower() + strzelec.getPower() +
jezdziec.getPower()));
```

```
        // update Archer price tooltip
        ui->pushButton_Buy_Archer->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                              "<ul>"
                                              "<li>Gold: " +
QString::number(strzelec.getPriceBuyGold()) + "</li>"
                                              "<li>Food: " +
QString::number(strzelec.getPriceBuyFood()) + "</li>"
                                              "<li>Iron: " +
QString::number(strzelec.getPriceBuyIron()) + "</li>"
                                              "</ul>");
    }

    // horseman
    else if ((button == ui->pushButton_Buy_Horseman)
            && ((piknier.getQuantity() + quantity + strzelec.getQuantity() +
jezdziec.getQuantity()) <= armia.getArmyLimit())
            && (zasoby.getGoldQuantity() - jezdziec.getPriceBuyGold() >= 0) &&
(zasoby.getFoodQuantity() - jezdziec.getPriceBuyFood() >= 0) &&
(zasoby.getIronQuantity() - jezdziec.getPriceBuyIron() >= 0))
    {
        jezdziec.buy(10, quantity, 30, 30, 30);

        // remove resources and update resource values
        zasoby.removeGold(jezdziec.getPriceBuyGold());
        zasoby.removeIron(jezdziec.getPriceBuyIron());
        zasoby.removeFood(jezdziec.getPriceBuyFood());

        // update jezdziec quantity and people quantity label
        ui->labelHorsemanQuantity->setText("Quantity: " +
QString::number(jezdziec.getQuantity()));
        ui->label_peopleValue->setText("People: " +
QString::number(piknier.getQuantity() + strzelec.getQuantity() +
jezdziec.getQuantity()));

        // update level label and resource value labels
        ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
        ui->label_ironValue->setText(QString::number(zasoby.getIronQuantity(),
'g', 15));
        ui->label_foodValue->setText(QString::number(zasoby.getFoodQuantity(),
'g', 15));

        // update army power label
        ui->labelArmyPower->setText("Army power: " +
QString::number(piknier.getPower() + strzelec.getPower() +
jezdziec.getPower()));

        // update jezdziec price tooltip
```

```
        ui->pushButton_Buy_Horseman->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                                "<ul>"
                                                "<li>Gold: " +
QString::number(jezdziec.getPriceBuyGold()) + "</li>"
                                                "<li>Food: " +
QString::number(jezdziec.getPriceBuyFood()) + "</li>"
                                                "<li>Iron: " +
QString::number(jezdziec.getPriceBuyIron()) + "</li>"
                                                "</ul>");
    }

    // open limit exceeded error
    else if (piknier.getQuantity() + quantity + strzelec.getQuantity() +
jezdziec.getQuantity() > armia.getArmyLimit())
    {
        LlimitExceededError limitExceededError;
        limitExceededError.setModal(true);
        limitExceededError.exec();
    }
    // open not enough resources error
    else
    {
        NoResourceError noResourceError;
        noResourceError.setModal(true);
        noResourceError.exec();
    }

}
//#####################################

//*************************************************************************
*******
//*************************************************************************
*******
//*************************************************************************
*******

//#####################################
// Function controlling "upgrade army" button
void MainWindow::upgradeArmyButtonPressed()
{
    QPushButton * button = (QPushButton*)sender();

    // pikeman
    if ((button == ui->pushButton_Upgrade_Pikeman)
            && (piknier.getLevel() < 6)
```

```
            && (zasoby.getGoldQuantity() - piknier.getPriceGold() >= 0) &&
(zasoby.getFoodQuantity() - piknier.getPriceFood() >= 0) &&
(zasoby.getIronQuantity() - piknier.getPriceIron() >= 0))
    {

        // remove resources and update resource values
        zasoby.removeGold(piknier.getPriceGold());
        zasoby.removeIron(piknier.getPriceIron());
        zasoby.removeFood(piknier.getPriceFood());

        piknier.upgrade(1.5, 50,50,50);

        // update piknier level label
        ui->labelPikemanLevel->setText("Level: " +
QString::number(piknier.getLevel(), 'g', 15));
        if(piknier.getLevel() == 6)
            ui->labelPikemanLevel->setText("Level: 6 (MAX)");

        // update tooltip
        ui->pushButton_Upgrade_Pikeman->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                                   "<ul>"
                                                   "<li>Gold: " +
QString::number(piknier.getPriceGold()) + "</li>"
                                                   "<li>Food: " +
QString::number(piknier.getPriceFood()) + "</li>"
                                                   "<li>Iron: " +
QString::number(piknier.getPriceIron()) + "</li>"
                                                   "</ul>");
    }

    // horseman
    else if ((button == ui->pushButton_Upgrade_Horseman)
            && (jezdziec.getLevel() < 6)
            && (zasoby.getGoldQuantity() - jezdziec.getPriceGold() >= 0) &&
(zasoby.getFoodQuantity() - jezdziec.getPriceFood() >= 0) &&
(zasoby.getIronQuantity() - jezdziec.getPriceIron() >= 0))
    {

        // remove resources and update resource values
        zasoby.removeGold(jezdziec.getPriceGold());
        zasoby.removeIron(jezdziec.getPriceIron());
        zasoby.removeFood(jezdziec.getPriceFood());

        jezdziec.upgrade(1.5, 150,150,150);

        // update jezdziec level label
        ui->label_HorsemanLevel->setText("Level: " +
QString::number(jezdziec.getLevel(), 'g', 15));
```

```
        if(jezdziec.getLevel() == 6)
            ui->label_HorsemanLevel->setText("Level: 6 (MAX)");

        // update tooltip
        ui->pushButton_Upgrade_Horseman->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                                    "<ul>"
                                                    "<li>Gold: " +
QString::number(jezdziec.getPriceGold()) + "</li>"
                                                    "<li>Food: " +
QString::number(jezdziec.getPriceFood()) + "</li>"
                                                    "<li>Iron: " +
QString::number(jezdziec.getPriceIron()) + "</li>"
                                                    "</ul>");
    }

    // Archer
    else if ((button == ui->pushButton_Upgrade_Archer)
            && (strzelec.getLevel() < 6)
            && (zasoby.getGoldQuantity() - strzelec.getPriceGold() >= 0) &&
(zasoby.getFoodQuantity() - strzelec.getPriceFood() >= 0) &&
(zasoby.getIronQuantity() - strzelec.getPriceIron() >= 0))
    {

        // remove resources and update resource values
        zasoby.removeGold(strzelec.getPriceGold());
        zasoby.removeIron(strzelec.getPriceIron());
        zasoby.removeFood(strzelec.getPriceFood());

        strzelec.upgrade(1.5, 100,100,100);

        // update strzelec level label
        ui->label_ArcherLevel->setText("Level: " +
QString::number(strzelec.getLevel(), 'g', 15));
        if(strzelec.getLevel() == 6)
            ui->label_ArcherLevel->setText("Level: 6 (MAX)");

        // update tooltip
        ui->pushButton_Upgrade_Archer->setToolTip("<font
color='red'><b>Cost:</b></font> "
                                                    "<ul>"
                                                    "<li>Gold: " +
QString::number(strzelec.getPriceGold()) + "</li>"
                                                    "<li>Food: " +
QString::number(strzelec.getPriceFood()) + "</li>"
                                                    "<li>Iron: " +
QString::number(strzelec.getPriceIron()) + "</li>"
                                                    "</ul>");
    }
```

```
    // open not enough resources error
    else
    {
        NoResourceError noResourceError;
        noResourceError.setModal(true);
        noResourceError.exec();
    }

    // update resource value labels
    ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
    ui->label_ironValue->setText(QString::number(zasoby.getIronQuantity(),
'g', 15));
    ui->label_foodValue->setText(QString::number(zasoby.getFoodQuantity(),
'g', 15));

    // if max level reached disable upgrade buttons
    if (piknier.getLevel() == 6)
    {
        ui->pushButton_Upgrade_Pikeman->setDisabled(true);
    }
    if (strzelec.getLevel() == 6)
    {
        ui->pushButton_Upgrade_Archer->setDisabled(true);
    }
    if (jezdziec.getLevel() == 6)
    {
        ui->pushButton_Upgrade_Horseman->setDisabled(true);
    }

    // update army power label
    ui->labelArmyPower->setText("Army power: " +
QString::number(piknier.getPower() + strzelec.getPower() +
jezdziec.getPower(), 'g', 15));
}
//#####################################

//*************************************************************************
*******
//*************************************************************************
*******
//*************************************************************************
*******

//#####################################
// function controlling attack enemy button

void MainWindow::attackEnemyButton()
```

```
{
    if ((piknier.getPower() + strzelec.getPower() + jezdziec.getPower()) >=
wrog.getPower())
    {
        // level 1
        if (ui->label_enemyLevel->text() == "Level: 1")
        {
            // add resources
            zasoby.addGold(wrog.getRewardGold());
            zasoby.addWood(wrog.getRewardWood());
            zasoby.addStone(wrog.getRewardStone());
            zasoby.addIron(wrog.getRewardIron());
            zasoby.addFood(wrog.getRewardFood());

            // update enemy values
            wrog.setRewardGold(1000);
            wrog.setRewardWood(1000);
            wrog.setRewardStone(1000);
            wrog.setRewardIron(1000);
            wrog.setRewardFood(1000);
            wrog.setPower(30);

            // update enemy labels
            ui->label_enemyLevel->setText("Level: 2");
            ui->label_enemyName->setText("Name: Gandalf");
            ui->label_enemyPower->setText("Power: " +
QString::number(wrog.getPower(), 'g', 15));
        }

        // level 2
        else if (ui->label_enemyLevel->text() == "Level: 2")
        {
            // add resources
            zasoby.addGold(wrog.getRewardGold());
            zasoby.addWood(wrog.getRewardWood());
            zasoby.addStone(wrog.getRewardStone());
            zasoby.addIron(wrog.getRewardIron());
            zasoby.addFood(wrog.getRewardFood());

            // update enemy values
            wrog.setRewardGold(1000);
            wrog.setRewardWood(1000);
            wrog.setRewardStone(1000);
            wrog.setRewardIron(1000);
            wrog.setRewardFood(1000);
            wrog.setPower(60);

            // update enemy labels
            ui->label_enemyLevel->setText("Level: 3");
```

```
            ui->label_enemyName->setText("Name: Geralt");
            ui->label_enemyPower->setText("Power: " +
QString::number(wrog.getPower(), 'g', 15));
        }

        // level 3
        else if (ui->label_enemyLevel->text() == "Level: 3")
        {
            // add resources
            zasoby.addGold(wrog.getRewardGold());
            zasoby.addWood(wrog.getRewardWood());
            zasoby.addStone(wrog.getRewardStone());
            zasoby.addIron(wrog.getRewardIron());
            zasoby.addFood(wrog.getRewardFood());

            // update enemy values
            wrog.setRewardGold(3000);
            wrog.setRewardWood(3000);
            wrog.setRewardStone(3000);
            wrog.setRewardIron(3000);
            wrog.setRewardFood(3000);
            wrog.setPower(90);

            // update enemy labels
            ui->label_enemyLevel->setText("Level: 4");
            ui->label_enemyName->setText("Name: Helga");
            ui->label_enemyPower->setText("Power: " +
QString::number(wrog.getPower(), 'g', 15));
        }

        // level 4
        else if (ui->label_enemyLevel->text() == "Level: 4")
        {
            // add resources
            zasoby.addGold(wrog.getRewardGold());
            zasoby.addWood(wrog.getRewardWood());
            zasoby.addStone(wrog.getRewardStone());
            zasoby.addIron(wrog.getRewardIron());
            zasoby.addFood(wrog.getRewardFood());

            ui->pushButtonAttackEnemy->setDisabled(true);

            // update enemy labels
            ui->label_enemyLevel->setText("Level: NONE");
            ui->label_enemyName->setText("Name: NONE");
            ui->label_enemyPower->setText("Power: NONE");

            ui->labelArmySection_2->setText("YOU HAVE DEFEATED ALL THE
ENEMIES");
```

```
        }

        // kill my units
        if(piknier.getQuantity() > 0)
        {
            int randomNum = static_cast<int>(qrand() % piknier.getQuantity());
            piknier.removeQuantity(randomNum);
            piknier.removePower(randomNum * 5);
        }

        if(strzelec.getQuantity() > 0)
        {
            int randomNum2 = static_cast<int>(qrand() %
strzelec.getQuantity());
            strzelec.removeQuantity(randomNum2);
            strzelec.removePower(randomNum2 * 7);
        }

        if(jezdziec.getQuantity() > 0)
        {
            int randomNum3 = static_cast<int>(qrand() %
jezdziec.getQuantity());
            jezdziec.removeQuantity(randomNum3);
            jezdziec.removePower(randomNum3 * 10);
        }

        // update resource labels
        ui->label_goldValue->setText(QString::number(zasoby.getGoldQuantity(),
'g', 15));
        ui->label_woodValue->setText(QString::number(zasoby.getWoodQuantity(),
'g', 15));
        ui->label_stoneValue-
>setText(QString::number(zasoby.getStoneQuantity(), 'g', 15));
        ui->label_ironValue->setText(QString::number(zasoby.getIronQuantity(),
'g', 15));
        ui->label_foodValue->setText(QString::number(zasoby.getFoodQuantity(),
'g', 15));
    }

    // if my army is weaker it DIES.
    else
    {
        // open up the error message
        youDied failure;
        failure.setModal(true);
        failure.exec();

        // kill ALL units
```

```
        piknier.setQuantity(0);
        strzelec.setQuantity(0);
        jezdziec.setQuantity(0);

        //  set army power to 0
        piknier.setPower(0);
        strzelec.setPower(0);
        jezdziec.setPower(0);
    }

    //update power label
    ui->labelArmyPower->setText("Army power: " +
QString::number(piknier.getPower() + strzelec.getPower() +
jezdziec.getPower(), 'g', 15));

    // update unit quantity labels
    ui->labelPikemanQuantity->setText("Quantity: " +
QString::number(piknier.getQuantity(), 'g', 15));
    ui->labelArcherQuantity->setText("Quantity: " +
QString::number(strzelec.getQuantity(), 'g', 15));
    ui->labelHorsemanQuantity->setText("Quantity: " +
QString::number(jezdziec.getQuantity(), 'g', 15));
    ui->label_peopleValue->setText("People: " +
QString::number(piknier.getQuantity() + strzelec.getQuantity() +
jezdziec.getQuantity(), 'g', 15));
}

//#####################################
```

### 6.1.2. Resource.cs

```
#include "resource.h"

//#####################################################
// CONSTRUCTORS
//#####################################################
Resource::Resource()
{
}

Resource::Resource(int gold, int wood, int stone, int iron, int food)
{
    setGoldQuantity(gold);
    setWoodQuantity(wood);
    setStoneQuantity(stone);
    setIronQuantity(iron);
    setFoodQuantity(food);
}
```

```
//#####################################################
// MEMBER FUNCTIONS
//#####################################################
// remove
void Resource::removeGold(int quantity)
{
    goldQuantity -= quantity;
}

void Resource::removeWood(int quantity)
{
    woodQuantity -= quantity;
}

void Resource::removeStone(int quantity)
{
    stoneQuantity -= quantity;
}

void Resource::removeIron(int quantity)
{
    ironQuantity -= quantity;
}

void Resource::removeFood(int quantity)
{
    foodQuantity -= quantity;
}

// add
void Resource::addGold(int quantity)
{
    goldQuantity += quantity;
}

void Resource::addWood(int quantity)
{
    woodQuantity += quantity;
}

void Resource::addStone(int quantity)
{
    stoneQuantity += quantity;
}

void Resource::addIron(int quantity)
{
    ironQuantity += quantity;
```

```
}

void Resource::addFood(int quantity)
{
    foodQuantity += quantity;
}

//#################################################
// SETTERS
//#################################################
void Resource::setGoldQuantity(int gold)
{
    goldQuantity = gold;
}

void Resource::setWoodQuantity(int wood)
{
    woodQuantity = wood;
}

void Resource::setStoneQuantity(int stone)
{
    stoneQuantity = stone;
}

void Resource::setIronQuantity(int iron)
{
    ironQuantity = iron;
}

void Resource::setFoodQuantity(int food)
{
    foodQuantity = food;
}

//#################################################
// GETTERS
//#################################################
int Resource::getGoldQuantity()
{
    return goldQuantity;
}
int Resource::getWoodQuantity()
{
    return woodQuantity;
}
int Resource::getStoneQuantity()
{
    return stoneQuantity;
```

```
}
int Resource::getIronQuantity()
{
    return ironQuantity;
}
int Resource::getFoodQuantity()
{
    return foodQuantity;
}
```

### 6.1.3. Price.cs

```
#include "price.h"

// CONSTRUCTORS
Price::Price()
{
}

// MEMBER FUNCTIONS
void Price::increasePrice(double gold, double wood, double stone, double iron,
double food)
{
    priceGold += gold;
    priceWood += wood;
    priceStone += stone;
    priceIron += iron;
    priceFood += food;
}


void Price::increaseBuyPrice(double gold, double wood, double stone, double
iron, double food)
{
    priceBuyGold += gold;
    priceBuyWood += wood;
    priceBuyStone += stone;
    priceBuyIron += iron;
    priceBuyFood += food;
}
// SETTERS
void Price::setPriceGold(int price)
{
    priceGold = price;
}

void Price::setPriceWood(int price)
{
```

```
    priceWood = price;
}

void Price::setPriceStone(int price)
{
    priceStone = price;
}

void Price::setPriceIron(int price)
{
    priceIron = price;
}

void Price::setPriceFood(int price)
{
    priceFood = price;
}


void Price::setPriceBuyGold(int price)
{
    priceBuyGold = price;
}

void Price::setPriceBuyWood(int price)
{
    priceBuyWood = price;
}

void Price::setPriceBuyStone(int price)
{
    priceBuyStone = price;
}

void Price::setPriceBuyIron(int price)
{
    priceBuyIron = price;
}

void Price::setPriceBuyFood(int price)
{
    priceBuyFood = price;
}

// GETTERS
double Price::getPriceGold()
{
    return priceGold;
}
```

```
double Price::getPriceWood()
{
    return priceWood;
}

double Price::getPriceStone()
{
    return priceStone;
}

double Price::getPriceIron()
{
    return priceIron;
}

double Price::getPriceFood()
{
    return priceFood;
}


double Price::getPriceBuyGold()
{
    return priceBuyGold;
}

double Price::getPriceBuyWood()
{
    return priceBuyWood;
}

double Price::getPriceBuyStone()
{
    return priceBuyStone;
}

double Price::getPriceBuyIron()
{
    return priceBuyIron;
}

double Price::getPriceBuyFood()
{
    return priceBuyFood;
}
```

### 6.1.4. Building.cs

```
#include "building.h"

//#####################################################
// CONSTRUCTORS
//#####################################################
Building::Building()
{

}

Building::Building(int priceGold, int priceWood, int priceStone, double
dropRate = 1.0, int level = 1)
{
    setPriceGold(priceGold);
    setPriceWood(priceWood);
    setPriceStone(priceStone);
    setDropRate(dropRate);
    setLevel(level);
}

//#####################################################
// MEMBER FUNCTIONS
//#####################################################

void Building::increaseDropRate(double value)
{
    buildingDropRate *= value;
}

void Building::upgrade(int gold, int wood, int stone, double dropRate)
{
    buildingLevel += 1;
    switch (buildingLevel)
    {
    case 1:
        increasePrice(gold, wood, stone, 0, 0);
        increaseDropRate(1);
        break;
    case 2:
        increasePrice(1.1 * gold, 1.1 * wood, 1.1 * stone, 0, 0);
        increaseDropRate(1.1 * dropRate);
        break;
    case 3:
        increasePrice(1.2 * gold, 1.2 * wood, 1.2 * stone, 0, 0);
        increaseDropRate(1.2 * dropRate);
        break;
    case 4:
        increasePrice(1.5 * gold, 1.5 * wood, 1.5 * stone, 0, 0);
```

```
        increaseDropRate(1.3);
        break;
    case 5:
        increasePrice(1.8 * gold, 1.8 * wood, 1.8 * stone, 0, 0);
        increaseDropRate(1.4);
        break;
    default:
        increasePrice(0, 0, 0, 0, 0);
        increaseDropRate(1);
    }
}

//#################################################
// SETTERS
//#################################################
void Building::setDropRate(double dropRate)
{
    buildingDropRate = dropRate;
}

void Building::setLevel(int level)
{
    buildingLevel = level;
}

//#################################################
// GETTERS
//#################################################
double Building::getDropRate()
{
    return buildingDropRate;
}

int Building::getLevel()
{
    return buildingLevel;
}
```

### 6.1.5. Army.cpp

```
#include "army.h"

//#################################################
// CONSTRUCTORS
//#################################################
Army::Army()
{
```

```
}

//#################################################
// MEMBER FUNCTIONS
//#################################################
void Army::addQuantity(int value)
{
    quantity += value;
}

void Army::removeQuantity(int value)
{
    quantity -= value;
}

void Army::addPower(int value)
{
    power += value;
}

void Army::removePower(int value)
{
    power -= value;
}

void Army::multiplyPower(double value)
{
    power *= value;
}

//#################################################
// SETTERS
//#################################################
void Army::setPower(int value)
{
    power = value;
}

void Army::setQuantity(int value)
{
    quantity = value;
}

void Army::setLevel(int value)
{
    level = value;
}

//#################################################
```

```
// GETTERS
//#################################################
double Army::getPower()
{
    return power;
}

int Army::getLevel()
{
    return level;
}

int Army::getQuantity()
{
    return quantity;
}
```

### 6.1.6. MyArmy.cpp

```
#include "myarmy.h"

//#################################################
// CONSTRUCTORS
//#################################################
MyArmy::MyArmy()
{

}

MyArmy::MyArmy(int limit)
{
    setArmyLimit(limit);
}

//#################################################
// MEMBER FUNCTIONS
//#################################################
void MyArmy::addArmyLimit(int value)
{
    armyLimit += value;
}


void MyArmy::buy(int value, int quantity, int gold, int iron, int food)
{
    addQuantity(quantity);
    addPower(value);
    increaseBuyPrice(gold, 0, 0, iron, food);
```

```
}

void MyArmy::upgrade(double powerRate, int gold, int food, int iron)
{
    level += 1;

    multiplyPower(powerRate);

    switch (level)
    {
    case 1:
        increasePrice(gold, 0, 0, iron, food);
        break;
    case 2:
        increasePrice(1.1 * gold, 0, 0, 1.1 * iron, 1.1 * food);
        break;
    case 3:
        increasePrice(1.2 * gold, 0, 0, 1.2 * iron, 1.2 * food);
        break;
    case 4:
        increasePrice(1.3 * gold, 0, 0, 1.3 * iron, 1.3 * food);
        break;
    case 5:
        increasePrice(1.4 * gold, 0, 0, 1.4 * iron, 1.4 * food);
        break;
    default:
        increasePrice(0, 0, 0, 0, 0);
    }
}


//#####################################################
// SETTERS
//#####################################################
void MyArmy::setArmyLimit(int value)
{
    armyLimit = value;
}

//#####################################################
// GETTERS
//#####################################################
int MyArmy::getArmyLimit()
{
    return armyLimit;
}
```

### 6.1.7. Enemy.cpp

```
#include "enemy.h"

//#####################################################
// CONSTRUCTORS
//#####################################################
Enemy::Enemy()
{

}

Enemy::Enemy(double enemyPower, int rewardGold, int rewardWood, int
rewardStone, int rewardIron, int rewardFood)
{
    setPower(enemyPower);
    setRewardGold(rewardGold);
    setRewardStone(rewardStone);
    setRewardWood(rewardWood);
    setRewardIron(rewardIron);
    setRewardFood(rewardFood);
}

//#####################################################
// MEMBER FUNCTIONS
//#####################################################

//#####################################################
// SETTERS
//#####################################################
void Enemy::setRewardGold(int value)
{
    rewardGold = value;
}

void Enemy::setRewardWood(int value)
{
    rewardWood = value;
}

void Enemy::setRewardStone(int value)
{
    rewardStone = value;
}

void Enemy::setRewardIron(int value)
{
    rewardIron = value;
}
```

```
void Enemy::setRewardFood(int value)
{
    rewardFood = value;
}

//#################################################
// GETTERS
//#################################################
int Enemy::getRewardGold()
{
    return rewardGold;
}

int Enemy::getRewardWood()
{
    return rewardWood;
}

int Enemy::getRewardStone()
{
    return rewardStone;
}

int Enemy::getRewardIron()
{
    return rewardIron;
}

int Enemy::getRewardFood()
{
    return rewardFood;
}
```

## 6.2. Listing kodu C#

### *6.2.1. MainWindows.xaml.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
```

```
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Windows.Threading;

namespace MiddleAges
{

    //#####################################################
    // POLYMORPHISM
    //#####################################################
    abstract public class Napis
    {
        abstract public string PobierzTekst();
    }

    public class Ciekawostka1 : Napis
    {
        public override string PobierzTekst()
        {
            return "Ciekawostka 1: Magowie używają magicznych zaklęć.";
            throw new NotImplementedException();
        }
    }

    public class Ciekawostka2 : Napis
    {
        public override string PobierzTekst()
        {
            return "Ciekawostka 2: Wojownicy są silni, a łucznicy strzelają";
            throw new NotImplementedException();
        }
    }

    /// <summary>
    /// Logika interakcji dla klasy MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        //#####################################################
        // OBJECTS DECLARATION
        //#####################################################
        Resource zasoby = new Resource(5000, 5000, 5000, 5000, 1000);
        Sawmill tartak = new Sawmill(300, 100, 30, 1.0, 1);
        StonePit kamieniolom = new StonePit(300, 100, 30, 1.0, 1);
        Farm farma = new Farm(200, 50, 50, 1.0, 1);
        IronMine kopalnia_zelaza = new IronMine(400, 100, 200, 1.0, 1);
        Hut domek = new Hut(50, 50, 50, 1.0, 1);
        Barracks koszary = new Barracks(500, 300, 400, 1.0, 1);
        MyArmy armia = new MyArmy(0);
```

```
        Pikeman piknier = new Pikeman(0, 0, 1, 10, 10, 10);
        Archer strzelec = new Archer(0, 0, 1, 20, 20, 20);
        Horseman jezdziec = new Horseman(0, 0, 1, 30, 30, 30);
        Enemy wrog = new Enemy(10, 500, 500, 500, 500, 500);
        DispatcherTimer goldTimer = new DispatcherTimer();
        DispatcherTimer woodTimer = new DispatcherTimer();
        DispatcherTimer stoneTimer = new DispatcherTimer();
        DispatcherTimer ironTimer = new DispatcherTimer();
        DispatcherTimer foodTimer = new DispatcherTimer();
        DispatcherTimer foodMinusTimer = new DispatcherTimer();
        DispatcherTimer ciekawostkaTimer = new DispatcherTimer();
        Random rnd = new Random();
        int goldDropValue = 0;
        int woodDropValue = 0;
        int stoneDropValue = 0;
        int ironDropValue = 0;
        int foodDropValue = 0;
        Napis c1 = new Ciekawostka1();
        Napis c2 = new Ciekawostka2();

        //#####################################################
        // Constructor
        //#####################################################
        public MainWindow()
        {
            InitializeComponent();

            //#####################################################
            // DEFAULT LABELS
            //#####################################################
            // Resource values
            label_GoldValue.Text = zasoby.GoldQuantity.ToString();
            label_WoodValue.Text = zasoby.WoodQuantity.ToString();
            label_StoneValue.Text = zasoby.StoneQuantity.ToString();
            label_IronValue.Text = zasoby.IronQuantity.ToString();
            label_FoodValue.Text = zasoby.FoodQuantity.ToString();

            // Sawmill
            label_SawmillLevel.Text = "Level: " +
tartak.BuildingLevel.ToString();
            label_SawmillGoldCost.Text = tartak.PriceUpgradeGold.ToString();
            label_SawmillWoodCost.Text = tartak.PriceUpgradeWood.ToString();
            label_SawmillStoneCost.Text = tartak.PriceUpgradeStone.ToString();

            // Stone Pit
            label_StonePitLevel.Text = "Level: " +
kamieniolom.BuildingLevel.ToString();
            label_StonePitGoldCost.Text =
kamieniolom.PriceUpgradeGold.ToString();
```

```
            label_StonePitWoodCost.Text =
kamieniolom.PriceUpgradeWood.ToString();
            label_StonePitStoneCost.Text =
kamieniolom.PriceUpgradeStone.ToString();

            // Iron Mine
            label_IronMineLevel.Text = "Level: " +
kopalnia_zelaza.BuildingLevel.ToString();
            label_IronMineGoldCost.Text =
kopalnia_zelaza.PriceUpgradeGold.ToString();
            label_IronMineWoodCost.Text =
kopalnia_zelaza.PriceUpgradeWood.ToString();
            label_IronMineStoneCost.Text =
kopalnia_zelaza.PriceUpgradeStone.ToString();

            // Farm
            label_FarmLevel.Text = "Level: " + farma.BuildingLevel.ToString();
            label_FarmGoldCost.Text = farma.PriceUpgradeGold.ToString();
            label_FarmWoodCost.Text = farma.PriceUpgradeWood.ToString();
            label_FarmStoneCost.Text = farma.PriceUpgradeStone.ToString();

            // Hut
            label_HutLevel.Text = "Level: " + domek.BuildingLevel.ToString();
            label_HutGoldCost.Text = domek.PriceUpgradeGold.ToString();
            label_HutWoodCost.Text = domek.PriceUpgradeWood.ToString();
            label_HutStoneCost.Text = domek.PriceUpgradeStone.ToString();

            // Barracks
            label_BarracksLevel.Text = "Level: " +
koszary.BuildingLevel.ToString();
            label_BarracksGoldCost.Text = koszary.PriceUpgradeGold.ToString();
            label_BarracksWoodCost.Text = koszary.PriceUpgradeWood.ToString();
            label_BarracksStoneCost.Text =
koszary.PriceUpgradeStone.ToString();

            // Army
            label_PikemanQuantity.Text = "Quantity: " +
piknier.Quantity.ToString();
            label_PikemanLevel.Text = "Level: " + piknier.Level.ToString();
            label_ArcherQuantity.Text = "Quantity: " +
strzelec.Quantity.ToString();
            label_ArcherLevel.Text = "Level: " + strzelec.Level.ToString();
            label_HorsemanQuantity.Text = "Quantity: " +
jezdziec.Quantity.ToString();
            label_HorsemanLevel.Text = "Level: " + jezdziec.Level.ToString();

            // disable army buttons
            button_PikemanBuy.IsEnabled = false;
            button_ArcherBuy.IsEnabled = false;
```

```
            button_HorsemanBuy.IsEnabled = false;
            button_PikemanUpgrade.IsEnabled = false;
            button_ArcherUpgrade.IsEnabled = false;
            button_HorsemanUpgrade.IsEnabled = false;

            // set timers
            goldTimer.Tick += goldTimer_Tick;
            goldTimer.Interval = new TimeSpan(0, 0, 1);
            goldTimer.Start();
            woodTimer.Tick += woodTimer_Tick;
            woodTimer.Interval = new TimeSpan(0, 0, 1);
            woodTimer.Start();
            stoneTimer.Tick += stoneTimer_Tick;
            stoneTimer.Interval = new TimeSpan(0, 0, 1);
            stoneTimer.Start();
            ironTimer.Tick += ironTimer_Tick;
            ironTimer.Interval = new TimeSpan(0, 0, 1);
            ironTimer.Start();
            foodTimer.Tick += foodTimer_Tick;
            foodTimer.Interval = new TimeSpan(0, 0, 1);
            foodTimer.Start();
            foodMinusTimer.Tick += foodMinusTimer_Tick;
            foodMinusTimer.Interval = new TimeSpan(0, 0, 10);
            foodMinusTimer.Start();
            ciekawostkaTimer.Tick += ciekawostkaTimer_Tick;
            ciekawostkaTimer.Interval = new TimeSpan(0, 0, 15);
            ciekawostkaTimer.Start();


            // army buy and upgrade tooltips
            button_PikemanBuy.ToolTip = "Gold: " +
piknier.PriceBuyGold.ToString() + " | Iron: " +
piknier.PriceBuyIron.ToString() + " | Food: " +
piknier.PriceBuyFood.ToString();
            button_ArcherBuy.ToolTip = "Gold: " +
strzelec.PriceBuyGold.ToString() + " | Iron: " +
strzelec.PriceBuyIron.ToString() + " | Food: " +
strzelec.PriceBuyFood.ToString();
            button_HorsemanBuy.ToolTip = "Gold: " +
jezdziec.PriceBuyGold.ToString() + " | Iron: " +
jezdziec.PriceBuyIron.ToString() + " | Food: " +
jezdziec.PriceBuyFood.ToString();
            button_PikemanUpgrade.ToolTip = "Gold: " +
piknier.PriceUpgradeGold.ToString() + " | Iron: " +
piknier.PriceUpgradeIron.ToString() + " | Food: " +
piknier.PriceUpgradeFood.ToString();
            button_ArcherUpgrade.ToolTip = "Gold: " +
strzelec.PriceUpgradeGold.ToString() + " | Iron: " +
```

```
strzelec.PriceUpgradeIron.ToString() + " | Food: " +
strzelec.PriceUpgradeFood.ToString();
            button_HorsemanUpgrade.ToolTip = "Gold: " +
jezdziec.PriceUpgradeGold.ToString() + " | Iron: " +
jezdziec.PriceUpgradeIron.ToString() + " | Food: " +
jezdziec.PriceUpgradeFood.ToString();

            // barracks level 1 tooltip
            button_BarracksUpgrade.ToolTip = "Unlocks PIKEMEN";

            // set the ciekawostka label to the first ciekawostka
            label_Ciekawostka.Content = c1.PobierzTekst();
        }

        //#####################################################
        // TIMERS
        //#####################################################
        private void goldTimer_Tick(object sender, EventArgs e)
        {
            zasoby.AddGold(goldDropValue);
            label_GoldValue.Text = zasoby.GoldQuantity.ToString();
        }

        private void woodTimer_Tick(object sender, EventArgs e)
        {
            zasoby.AddWood(woodDropValue);
            label_WoodValue.Text = zasoby.WoodQuantity.ToString();
        }

        private void stoneTimer_Tick(object sender, EventArgs e)
        {
            zasoby.AddStone(stoneDropValue);
            label_StoneValue.Text = zasoby.StoneQuantity.ToString();
        }

        private void ironTimer_Tick(object sender, EventArgs e)
        {
            zasoby.AddIron(ironDropValue);
            label_IronValue.Text = zasoby.IronQuantity.ToString();
        }

        private void foodTimer_Tick(object sender, EventArgs e)
        {
            zasoby.AddFood(foodDropValue);
            label_FoodValue.Text = zasoby.FoodQuantity.ToString();
        }

        private void ciekawostkaTimer_Tick(object sender, EventArgs e)
        {
```

```
            if (label_Ciekawostka.Content == c1.PobierzTekst())
                label_Ciekawostka.Content = c2.PobierzTekst();
            else
                label_Ciekawostka.Content = c1.PobierzTekst();
        }

        private void foodMinusTimer_Tick(object sender, EventArgs e)
        {
            int x = piknier.Quantity + jezdziec.Quantity + strzelec.Quantity;
            if (x > 0)
            {
                if (zasoby.FoodQuantity - x >= 0)
                {
                    for (int i = 0; i < x; i++)
                        zasoby.RemoveFood(1);

                    label_FoodValue.Text = zasoby.FoodQuantity.ToString();

                    // show up the tax message
                    MessageBox.Show("Food tax taken. " + x + " food
removed.");
                }
                else
                    desertion();
            }
        }

        // function that controlls army desertion when player has not enough
food
        private void desertion()
        {
            int randomNumber = rnd.Next(piknier.Quantity);
            int randomNumber2 = rnd.Next(strzelec.Quantity);
            int randomNumber3 = rnd.Next(jezdziec.Quantity);

            if (piknier.Quantity > 0)
            {
                piknier.RemoveQuantity(randomNumber);
                piknier.RemovePower(randomNumber * 5);
            }
            if (strzelec.Quantity > 0)
            {
                strzelec.RemoveQuantity(randomNumber2);
                strzelec.RemovePower(randomNumber2 * 5);
            }
            if (jezdziec.Quantity > 0)
            {
                jezdziec.RemoveQuantity(randomNumber3);
                jezdziec.RemovePower(randomNumber3 * 5);
```

```
            }

            // update labels
            label_ArmyPower.Text = "Army power: " + (piknier.Power +
strzelec.Power + jezdziec.Power).ToString();
            label_PikemanQuantity.Text = "Quantity: " +
piknier.Quantity.ToString();
            label_ArcherQuantity.Text = "Quantity: " +
strzelec.Quantity.ToString();
            label_HorsemanQuantity.Text = "Quantity: " +
jezdziec.Quantity.ToString();
            label_peopleValue.Text = (piknier.Quantity + strzelec.Quantity +
jezdziec.Quantity).ToString();

            // show up desertion message
            MessageBox.Show("People are starving!\nSoldiers have deserted.\n"
+
                randomNumber + " pikemen, " +
                randomNumber2 + " archers, " +
                randomNumber3 + " horsemen run away.");
        }

        //#################################################
        // BUILDING UPGRADE BUTTONS
        //#################################################
        // sawmill upgrade button
        private void button_SawmillUpgrade_Click(object sender,
RoutedEventArgs e)
        {
            if (tartak.BuildingLevel < 6 && zasoby.GoldQuantity -
tartak.PriceUpgradeGold >= 0 && zasoby.WoodQuantity - tartak.PriceUpgradeWood
>= 0 && zasoby.StoneQuantity - tartak.PriceUpgradeStone >= 0)
            {
                // increase the drop value
                woodDropValue++;

                // remove required resources
                zasoby.RemoveGold((int)tartak.PriceUpgradeGold);
                zasoby.RemoveWood((int)tartak.PriceUpgradeWood);
                zasoby.RemoveStone((int)tartak.PriceUpgradeStone);

                // upgrade building
                tartak.Upgrade(50, 30, 30);

                //update labels
                label_SawmillLevel.Text = "Level: " +
tartak.BuildingLevel.ToString();
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_WoodValue.Text = zasoby.WoodQuantity.ToString();
```

```
                label_StoneValue.Text = zasoby.StoneQuantity.ToString();
                label_WoodIncome.Text = "+ " + woodDropValue.ToString() +
"/sec";
                if (label_SawmillLevel.Text == "Level: 6")
                {
                    label_SawmillGoldCost.Text = "--";
                    label_SawmillWoodCost.Text = "--";
                    label_SawmillStoneCost.Text = "--";
                }
                else
                {
                    label_SawmillGoldCost.Text =
tartak.PriceUpgradeGold.ToString();
                    label_SawmillWoodCost.Text =
tartak.PriceUpgradeWood.ToString();
                    label_SawmillStoneCost.Text =
tartak.PriceUpgradeStone.ToString();
                }

                // disable upgrade button when level == 6
                if (tartak.BuildingLevel == 6)
                {
                    button_SawmillUpgrade.IsEnabled = false;
                }
            }
            else
                // show error dialog window
                MessageBox.Show("Not enough resources!");
        }

        // farm upgrade button
        private void button_FarmUpgrade_Click(object sender, RoutedEventArgs
e)
        {
            if (farma.BuildingLevel < 6 && zasoby.GoldQuantity -
farma.PriceUpgradeGold >= 0 && zasoby.WoodQuantity - farma.PriceUpgradeWood >=
0 && zasoby.StoneQuantity - farma.PriceUpgradeStone >= 0)
            {
                // increase the drop value
                foodDropValue++;

                // remove required resources
                zasoby.RemoveGold((int)farma.PriceUpgradeGold);
                zasoby.RemoveWood((int)farma.PriceUpgradeWood);
                zasoby.RemoveStone((int)farma.PriceUpgradeStone);

                // upgrade building
                farma.Upgrade(50, 30, 30);
```

```
                //update labels
                label_FarmLevel.Text = "Level: " +
farma.BuildingLevel.ToString();
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_WoodValue.Text = zasoby.WoodQuantity.ToString();
                label_StoneValue.Text = zasoby.StoneQuantity.ToString();
                label_FoodIncome.Text = "+ " + foodDropValue.ToString() +
"/sec";
                if (label_FarmLevel.Text == "Level: 6")
                {
                    label_FarmGoldCost.Text = "--";
                    label_FarmWoodCost.Text = "--";
                    label_FarmStoneCost.Text = "--";
                }
                else
                {
                    label_FarmGoldCost.Text =
farma.PriceUpgradeGold.ToString();
                    label_FarmWoodCost.Text =
farma.PriceUpgradeWood.ToString();
                    label_FarmStoneCost.Text =
farma.PriceUpgradeStone.ToString();
                }

                // disable upgrade button when level == 6
                if (farma.BuildingLevel == 6)
                {
                    button_FarmUpgrade.IsEnabled = false;
                }
            }
            else
                // show error dialog window
                MessageBox.Show("Not enough resources!");
        }

        // stone pit upgrade
        private void button_StonePitUpgrade_Click(object sender,
RoutedEventArgs e)
        {
            if (kamieniolom.BuildingLevel < 6 && zasoby.GoldQuantity -
kamieniolom.PriceUpgradeGold >= 0 && zasoby.WoodQuantity -
kamieniolom.PriceUpgradeWood >= 0 && zasoby.StoneQuantity -
kamieniolom.PriceUpgradeStone >= 0)
            {
                // increase the drop value
                stoneDropValue++;

                // remove required resources
                zasoby.RemoveGold((int)kamieniolom.PriceUpgradeGold);
```

```
                zasoby.RemoveWood((int)kamieniolom.PriceUpgradeWood);
                zasoby.RemoveStone((int)kamieniolom.PriceUpgradeStone);

                // upgrade building
                kamieniolom.Upgrade(50, 30, 30);

                //update labels
                label_StonePitLevel.Text = "Level: " +
kamieniolom.BuildingLevel.ToString();
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_WoodValue.Text = zasoby.WoodQuantity.ToString();
                label_StoneValue.Text = zasoby.StoneQuantity.ToString();
                label_StoneIncome.Text = "+ " + stoneDropValue.ToString() +
"/sec";
                if (label_StonePitLevel.Text == "Level: 6")
                {
                    label_StonePitGoldCost.Text = "--";
                    label_StonePitWoodCost.Text = "--";
                    label_StonePitStoneCost.Text = "--";
                }
                else
                {
                    label_StonePitGoldCost.Text =
kamieniolom.PriceUpgradeGold.ToString();
                    label_StonePitWoodCost.Text =
kamieniolom.PriceUpgradeWood.ToString();
                    label_StonePitStoneCost.Text =
kamieniolom.PriceUpgradeStone.ToString();
                }

                // disable upgrade button when level == 6
                if (kamieniolom.BuildingLevel == 6)
                {
                    button_StonePitUpgrade.IsEnabled = false;
                }
            }
            else
                // show error dialog window
                MessageBox.Show("Not enough resources!");
        }

        // iron mine upgrade button
        private void button_IronMineUpgrade_Click(object sender,
RoutedEventArgs e)
        {
            if (kopalnia_zelaza.BuildingLevel < 6 && zasoby.GoldQuantity -
kopalnia_zelaza.PriceUpgradeGold >= 0 && zasoby.WoodQuantity -
kopalnia_zelaza.PriceUpgradeWood >= 0 && zasoby.StoneQuantity -
kopalnia_zelaza.PriceUpgradeStone >= 0)
```

```
            {
                // increase the drop value
                ironDropValue++;

                // remove required resources
                zasoby.RemoveGold((int)kopalnia_zelaza.PriceUpgradeGold);
                zasoby.RemoveWood((int)kopalnia_zelaza.PriceUpgradeWood);
                zasoby.RemoveStone((int)kopalnia_zelaza.PriceUpgradeStone);

                // upgrade building
                kopalnia_zelaza.Upgrade(50, 30, 30);

                //update labels
                label_IronMineLevel.Text = "Level: " +
kopalnia_zelaza.BuildingLevel.ToString();
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_WoodValue.Text = zasoby.WoodQuantity.ToString();
                label_StoneValue.Text = zasoby.StoneQuantity.ToString();
                label_IronIncome.Text = "+ " + ironDropValue.ToString() +
"/sec";
                if (label_IronMineLevel.Text == "Level: 6")
                {
                    label_IronMineGoldCost.Text = "--";
                    label_IronMineWoodCost.Text = "--";
                    label_IronMineStoneCost.Text = "--";
                }
                else
                {
                    label_IronMineGoldCost.Text =
kopalnia_zelaza.PriceUpgradeGold.ToString();
                    label_IronMineWoodCost.Text =
kopalnia_zelaza.PriceUpgradeWood.ToString();
                    label_IronMineStoneCost.Text =
kopalnia_zelaza.PriceUpgradeStone.ToString();
                }

                // disable upgrade button when level == 6
                if (kopalnia_zelaza.BuildingLevel == 6)
                {
                    button_IronMineUpgrade.IsEnabled = false;
                }
            }
            else
                // show error dialog window
                MessageBox.Show("Not enough resources!");
        }

        // hut upgrade button
        private void button_HutUpgrade_Click(object sender, RoutedEventArgs e)
```

```
        {
            if (domek.BuildingLevel < 6 && zasoby.GoldQuantity -
domek.PriceUpgradeGold >= 0 && zasoby.WoodQuantity - domek.PriceUpgradeWood >=
0 && zasoby.StoneQuantity - domek.PriceUpgradeStone >= 0)
            {
                // increase the drop value
                goldDropValue++;

                // remove required resources
                zasoby.RemoveGold((int)domek.PriceUpgradeGold);
                zasoby.RemoveWood((int)domek.PriceUpgradeWood);
                zasoby.RemoveStone((int)domek.PriceUpgradeStone);

                // upgrade building
                domek.Upgrade(50, 30, 30);
                armia.AddArmyLimit(10);

                //update labels
                label_HutLevel.Text = "Level: " +
domek.BuildingLevel.ToString();
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_WoodValue.Text = zasoby.WoodQuantity.ToString();
                label_StoneValue.Text = zasoby.StoneQuantity.ToString();
                label_GoldIncome.Text = "+ " + goldDropValue.ToString() +
"/sec";
                if (label_HutLevel.Text == "Level: 6")
                {
                    label_HutGoldCost.Text = "--";
                    label_HutWoodCost.Text = "--";
                    label_HutStoneCost.Text = "--";
                }
                else
                {
                    label_HutGoldCost.Text =
domek.PriceUpgradeGold.ToString();
                    label_HutWoodCost.Text =
domek.PriceUpgradeWood.ToString();
                    label_HutStoneCost.Text =
domek.PriceUpgradeStone.ToString();
                }
                label_maxPeopleValue.Text = "Limit: " +
armia.ArmyLimit.ToString();

                // disable upgrade button when level == 6
                if (domek.BuildingLevel == 6)
                {
                    button_HutUpgrade.IsEnabled = false;
                }
            }
```

```
            else
                // show error dialog window
                MessageBox.Show("Not enough resources!");
        }

        // barracks update button
        private void button_BarracksUpgrade_Click(object sender,
RoutedEventArgs e)
        {

            if (koszary.BuildingLevel < 4 && zasoby.GoldQuantity -
koszary.PriceUpgradeGold >= 0 && zasoby.WoodQuantity -
koszary.PriceUpgradeWood >= 0 && zasoby.StoneQuantity -
koszary.PriceUpgradeStone >= 0)
            {
                // update tooltips and enable army buttons
                if (koszary.BuildingLevel == 1)
                {
                    button_BarracksUpgrade.ToolTip = "Unlocks ARCHERS";
                    button_PikemanBuy.IsEnabled = true;
                    button_PikemanUpgrade.IsEnabled = true;
                }
                else if (koszary.BuildingLevel == 2)
                {
                    button_BarracksUpgrade.ToolTip = "Unlocks HORSEMEN";
                    button_ArcherBuy.IsEnabled = true;
                    button_ArcherUpgrade.IsEnabled = true;
                }
                else if (koszary.BuildingLevel == 3)
                {
                    button_HorsemanBuy.IsEnabled = true;
                    button_HorsemanUpgrade.IsEnabled = true;
                }

                // remove required resources
                zasoby.RemoveGold((int)koszary.PriceUpgradeGold);
                zasoby.RemoveWood((int)koszary.PriceUpgradeWood);
                zasoby.RemoveStone((int)koszary.PriceUpgradeStone);

                // upgrade building
                koszary.Upgrade(50, 30, 30);

                //update labels
                label_BarracksLevel.Text = "Level: " +
koszary.BuildingLevel.ToString();
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_WoodValue.Text = zasoby.WoodQuantity.ToString();
                label_StoneValue.Text = zasoby.StoneQuantity.ToString();
```

```
                label_WoodIncome.Text = "+ " + woodDropValue.ToString() +
"/sec";
                if (label_BarracksLevel.Text == "Level: 4")
                {
                    label_BarracksGoldCost.Text = "--";
                    label_BarracksWoodCost.Text = "--";
                    label_BarracksStoneCost.Text = "--";
                }
                else
                {
                    label_BarracksGoldCost.Text =
koszary.PriceUpgradeGold.ToString();
                    label_BarracksWoodCost.Text =
koszary.PriceUpgradeWood.ToString();
                    label_BarracksStoneCost.Text =
koszary.PriceUpgradeStone.ToString();
                }

                // disable upgrade button when level == 6
                if (koszary.BuildingLevel == 4)
                {
                    button_BarracksUpgrade.IsEnabled = false;
                }
            }
            else
                // show error dialog window
                MessageBox.Show("Not enough resources!");
        }

        //#################################################
        // ARMY BUY BUTTONS
        //#################################################
        // pikeman buy button
        private void button_PikemanBuy_Click(object sender, RoutedEventArgs e)
        {
            int quantity = 1;

            if (piknier.Quantity + strzelec.Quantity + jezdziec.Quantity +
quantity <= armia.ArmyLimit
                && zasoby.GoldQuantity - piknier.PriceBuyGold >= 0 &&
zasoby.IronQuantity - piknier.PriceBuyIron >= 0 && zasoby.FoodQuantity -
piknier.PriceBuyFood >= 0)
            {
                piknier.Buy(5, quantity, 10, 10, 10);

                // remove resources
                zasoby.RemoveGold((int)piknier.PriceBuyGold);
                zasoby.RemoveIron((int)piknier.PriceBuyIron);
                zasoby.RemoveFood((int)piknier.PriceBuyFood);
```

```
                // update labels
                label_PikemanQuantity.Text = "Quantity: " +
piknier.Quantity.ToString();
                int x = piknier.Quantity + strzelec.Quantity +
jezdziec.Quantity;
                label_peopleValue.Text = x.ToString();
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_IronValue.Text = zasoby.IronQuantity.ToString();
                label_FoodValue.Text = zasoby.FoodQuantity.ToString();
                label_ArmyPower.Text = "Army power: " + (piknier.Power +
strzelec.Power + jezdziec.Power).ToString();

                // update tooltip
                button_PikemanBuy.ToolTip = "Gold: " +
piknier.PriceBuyGold.ToString() + " | Iron: " +
piknier.PriceBuyIron.ToString() + " | Food: " +
piknier.PriceBuyFood.ToString();
            }
            else if (piknier.Quantity + strzelec.Quantity + jezdziec.Quantity
+ quantity > armia.ArmyLimit)
                MessageBox.Show("Army limit exceeded!");
            else
                MessageBox.Show("Not enough resources!");
        }

        // archer buy button
        private void button_ArcherBuy_Click(object sender, RoutedEventArgs e)
        {
            int quantity = 1;

            if (strzelec.Quantity + strzelec.Quantity + jezdziec.Quantity +
quantity <= armia.ArmyLimit
                && zasoby.GoldQuantity - strzelec.PriceBuyGold >= 0 &&
zasoby.IronQuantity - strzelec.PriceBuyIron >= 0 && zasoby.FoodQuantity -
strzelec.PriceBuyFood >= 0)
            {
                strzelec.Buy(5, quantity, 20, 20, 20);

                // remove resources
                zasoby.RemoveGold((int)strzelec.PriceBuyGold);
                zasoby.RemoveIron((int)strzelec.PriceBuyIron);
                zasoby.RemoveFood((int)strzelec.PriceBuyFood);

                // update labels
                label_ArcherQuantity.Text = "Quantity: " +
strzelec.Quantity.ToString();
                int x = piknier.Quantity + strzelec.Quantity +
jezdziec.Quantity;
```

```
                label_peopleValue.Text = x.ToString();
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_IronValue.Text = zasoby.IronQuantity.ToString();
                label_FoodValue.Text = zasoby.FoodQuantity.ToString();
                label_ArmyPower.Text = "Army power: " + (piknier.Power +
strzelec.Power + jezdziec.Power).ToString();

                // update tooltip
                button_ArcherBuy.ToolTip = "Gold: " +
strzelec.PriceBuyGold.ToString() + " | Iron: " +
strzelec.PriceBuyIron.ToString() + " | Food: " +
strzelec.PriceBuyFood.ToString();
            }
            else if (piknier.Quantity + strzelec.Quantity + jezdziec.Quantity
+ quantity > armia.ArmyLimit)
                MessageBox.Show("Army limit exceeded!");
            else
                MessageBox.Show("Not enough resources!");
        }

        // horseman buy button
        private void button_HorsemanBuy_Click(object sender, RoutedEventArgs
e)
        {
            int quantity = 1;

            if (jezdziec.Quantity + strzelec.Quantity + jezdziec.Quantity +
quantity <= armia.ArmyLimit
                && zasoby.GoldQuantity - jezdziec.PriceBuyGold >= 0 &&
zasoby.IronQuantity - jezdziec.PriceBuyIron >= 0 && zasoby.FoodQuantity -
jezdziec.PriceBuyFood >= 0)
            {
                jezdziec.Buy(5, quantity, 30, 30, 30);

                // remove resources
                zasoby.RemoveGold((int)jezdziec.PriceBuyGold);
                zasoby.RemoveIron((int)jezdziec.PriceBuyIron);
                zasoby.RemoveFood((int)jezdziec.PriceBuyFood);

                // update labels
                label_HorsemanQuantity.Text = "Quantity: " +
jezdziec.Quantity.ToString();
                int x = piknier.Quantity + strzelec.Quantity +
jezdziec.Quantity;
                label_peopleValue.Text = x.ToString();
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_IronValue.Text = zasoby.IronQuantity.ToString();
                label_FoodValue.Text = zasoby.FoodQuantity.ToString();
```

```
                label_ArmyPower.Text = "Army power: " + (piknier.Power +
strzelec.Power + jezdziec.Power).ToString();

                // update tooltip
                button_HorsemanBuy.ToolTip = "Gold: " +
jezdziec.PriceBuyGold.ToString() + " | Iron: " +
jezdziec.PriceBuyIron.ToString() + " | Food: " +
jezdziec.PriceBuyFood.ToString();
            }
            else if (piknier.Quantity + strzelec.Quantity + jezdziec.Quantity
+ quantity > armia.ArmyLimit)
                MessageBox.Show("Army limit exceeded!");
            else
                MessageBox.Show("Not enough resources!");
        }

        //#################################################
        // ARMY UPGRADE BUTTONS
        //#################################################
        // pikeman upgrade button
        private void button_PikemanUpgrade_Click(object sender,
RoutedEventArgs e)
        {
            if (piknier.Level < 6
                && zasoby.GoldQuantity - piknier.PriceUpgradeGold >= 0
                && zasoby.IronQuantity - piknier.PriceUpgradeIron >= 0
                && zasoby.FoodQuantity - piknier.PriceUpgradeFood >= 0)
            {
                // remove resources
                zasoby.RemoveGold((int)piknier.PriceUpgradeGold);
                zasoby.RemoveIron((int)piknier.PriceUpgradeIron);
                zasoby.RemoveFood((int)piknier.PriceUpgradeFood);

                piknier.Upgrade(1.5, 50, 50, 50);

                //update labels
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_IronValue.Text = zasoby.IronQuantity.ToString();
                label_FoodValue.Text = zasoby.FoodQuantity.ToString();
                label_ArmyPower.Text = "Army power: " + (piknier.Power +
strzelec.Power + jezdziec.Power).ToString();
                label_PikemanLevel.Text = "Level: " +
piknier.Level.ToString();

                //update tooltip
                button_PikemanUpgrade.ToolTip = "Gold: " +
piknier.PriceUpgradeGold.ToString() + " | Iron: " +
piknier.PriceUpgradeIron.ToString() + " | Food: " +
piknier.PriceUpgradeFood.ToString();
```

```
            }
            else
                MessageBox.Show("Not enough resources!");

            if (piknier.Level == 6)
                button_PikemanUpgrade.IsEnabled = false;
        }

        // archer upgrade button
        private void button_ArcherUpgrade_Click(object sender, RoutedEventArgs
e)
        {
            if (strzelec.Level < 6
                && zasoby.GoldQuantity - strzelec.PriceUpgradeGold >= 0
                && zasoby.IronQuantity - strzelec.PriceUpgradeIron >= 0
                && zasoby.FoodQuantity - strzelec.PriceUpgradeFood >= 0)
            {
                // remove resources
                zasoby.RemoveGold((int)strzelec.PriceUpgradeGold);
                zasoby.RemoveIron((int)strzelec.PriceUpgradeIron);
                zasoby.RemoveFood((int)strzelec.PriceUpgradeFood);

                strzelec.Upgrade(1.5, 50, 50, 50);

                //update labels
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_IronValue.Text = zasoby.IronQuantity.ToString();
                label_FoodValue.Text = zasoby.FoodQuantity.ToString();
                label_ArmyPower.Text = "Army power: " + (piknier.Power +
strzelec.Power + jezdziec.Power).ToString();
                label_ArcherLevel.Text = "Level: " +
strzelec.Level.ToString();

                //update tooltip
                button_ArcherUpgrade.ToolTip = "Gold: " +
strzelec.PriceUpgradeGold.ToString() + " | Iron: " +
strzelec.PriceUpgradeIron.ToString() + " | Food: " +
strzelec.PriceUpgradeFood.ToString();
            }
            else
                MessageBox.Show("Not enough resources!");

            if (strzelec.Level == 6)
                button_ArcherUpgrade.IsEnabled = false;
        }

        // archer upgrade button
        private void button_HorsemanUpgrade_Click(object sender,
RoutedEventArgs e)
```

```
        {
            if (jezdziec.Level < 6
                && zasoby.GoldQuantity - jezdziec.PriceUpgradeGold >= 0
                && zasoby.IronQuantity - jezdziec.PriceUpgradeIron >= 0
                && zasoby.FoodQuantity - jezdziec.PriceUpgradeFood >= 0)
            {
                // remove resources
                zasoby.RemoveGold((int)jezdziec.PriceUpgradeGold);
                zasoby.RemoveIron((int)jezdziec.PriceUpgradeIron);
                zasoby.RemoveFood((int)jezdziec.PriceUpgradeFood);

                jezdziec.Upgrade(1.5, 50, 50, 50);

                //update labels
                label_GoldValue.Text = zasoby.GoldQuantity.ToString();
                label_IronValue.Text = zasoby.IronQuantity.ToString();
                label_FoodValue.Text = zasoby.FoodQuantity.ToString();
                label_ArmyPower.Text = "Army power: " + (piknier.Power +
strzelec.Power + jezdziec.Power).ToString();
                label_HorsemanLevel.Text = "Level: " +
jezdziec.Level.ToString();

                //update tooltip
                button_HorsemanUpgrade.ToolTip = "Gold: " +
jezdziec.PriceUpgradeGold.ToString() + " | Iron: " +
jezdziec.PriceUpgradeIron.ToString() + " | Food: " +
jezdziec.PriceUpgradeFood.ToString();
            }
            else
                MessageBox.Show("Not enough resources!");

            if (jezdziec.Level == 6)
                button_HorsemanUpgrade.IsEnabled = false;
        }

        //#################################################
        // ENEMY INTERACTION BUTTONS
        //#################################################
        // attack enemy button
        private void button_EnemyAttack_Click(object sender, RoutedEventArgs
e)
        {
            if (piknier.Power + strzelec.Power + jezdziec.Power >= wrog.Power)
            {
                // add resources as a reward
                zasoby.AddGold(wrog.RewardGold);
                zasoby.AddWood(wrog.RewardFood);
                zasoby.AddStone(wrog.RewardStone);
                zasoby.AddIron(wrog.RewardIron);
```

```
                zasoby.AddFood(wrog.RewardFood);

                // enemy level 1
                if (label_EnemyLevel.Text == "Level: 1")
                {
                    //update enemy values
                    wrog.RewardGold = 1000;
                    wrog.RewardWood = 1000;
                    wrog.RewardStone = 1000;
                    wrog.RewardIron = 1000;
                    wrog.RewardFood = 1000;
                    wrog.Power = 30;

                    // update enemy labels
                    label_EnemyLevel.Text = "Level: 2";
                    label_EnemyName.Text = "Name: Gandalf";
                    label_EnemyPower.Text = "Power: " + wrog.Power.ToString();
                }

                // enemy level 2
                else if (label_EnemyLevel.Text == "Level: 2")
                {
                    //update enemy values
                    wrog.RewardGold = 2000;
                    wrog.RewardWood = 2000;
                    wrog.RewardStone = 2000;
                    wrog.RewardIron = 2000;
                    wrog.RewardFood = 2000;
                    wrog.Power = 60;

                    // update enemy labels
                    label_EnemyLevel.Text = "Level: 3";
                    label_EnemyName.Text = "Name: Geralt";
                    label_EnemyPower.Text = "Power: " + wrog.Power.ToString();
                }

                // enemy level 3
                else if (label_EnemyLevel.Text == "Level: 3")
                {
                    //update enemy values
                    wrog.RewardGold = 3000;
                    wrog.RewardWood = 3000;
                    wrog.RewardStone = 3000;
                    wrog.RewardIron = 3000;
                    wrog.RewardFood = 3000;
                    wrog.Power = 90;

                    // update enemy labels
                    label_EnemyLevel.Text = "Level: 4";
```

```
                label_EnemyName.Text = "Name: Helga";
                label_EnemyPower.Text = "Power: " + wrog.Power.ToString();
            }

            // enemy level 4
            else if (label_EnemyLevel.Text == "Level: 4")
            {
                // disable attack button
                button_EnemyAttack.IsEnabled = false;

                // update enemy labels
                label_EnemyLevel.Text = "Level: NONE";
                label_EnemyName.Text = "Name: NONE";
                label_EnemyPower.Text = "Power: NONE";

                label_ArmySection.Text = "YOU HAVE DEFEATED ALL THE
ENEMIES";

            }
            // kill my units
            int randomNumber = rnd.Next(piknier.Quantity);
            int randomNumber2 = rnd.Next(strzelec.Quantity);
            int randomNumber3 = rnd.Next(jezdziec.Quantity);

            if (piknier.Quantity > 0)
            {
                piknier.RemoveQuantity(randomNumber);
                piknier.RemovePower(randomNumber * 5);
            }
            if (strzelec.Quantity > 0)
            {
                strzelec.RemoveQuantity(randomNumber2);
                strzelec.RemovePower(randomNumber * 5);
            }
            if (jezdziec.Quantity > 0)
            {
                jezdziec.RemoveQuantity(randomNumber3);
                jezdziec.RemovePower(randomNumber * 5);
            }

            MessageBox.Show("You won!\n" +
                randomNumber + " pikemen, " +
                randomNumber2 + " archers, " +
                randomNumber3 + " horsemen died.");
        }

        // if my army is weaker all my soldiers die
        else
        {
```

```
                // show up the error message
                MessageBox.Show("You lost!\n All your soldiers died!");

                // kill all the soldiers
                piknier.Quantity = 0;
                strzelec.Quantity = 0;
                jezdziec.Quantity = 0;

                // set army power to 0
                piknier.Power = 0;
                strzelec.Power = 0;
                jezdziec.Power = 0;
            }

            // update resource labels
            label_GoldValue.Text = zasoby.GoldQuantity.ToString();
            label_WoodValue.Text = zasoby.WoodQuantity.ToString();
            label_StoneValue.Text = zasoby.StoneQuantity.ToString();
            label_IronValue.Text = zasoby.IronQuantity.ToString();
            label_FoodValue.Text = zasoby.FoodQuantity.ToString();

            //update power label and unit quantity labels
            label_ArmyPower.Text = "Army power: " + (piknier.Power +
strzelec.Power + jezdziec.Power).ToString();
            label_PikemanQuantity.Text = "Quantity: " +
piknier.Quantity.ToString();
            label_ArcherQuantity.Text = "Quantity: " +
strzelec.Quantity.ToString();
            label_HorsemanQuantity.Text = "Quantity: " +
jezdziec.Quantity.ToString();
            label_peopleValue.Text = (piknier.Quantity + strzelec.Quantity +
jezdziec.Quantity).ToString();
        }

        // credits button
        private void button_Credits_Click(object sender, RoutedEventArgs e)
        {
            MessageBox.Show("Code, idea: Szymon Wiśniewski\n" +
                "Icons: https://www.flaticon.com/authors/smashicons\n" +
                "https://www.flaticon.com/authors/freepik\n" +
                "https://www.flaticon.com/authors/roundicons\n" +
                "Building's images: Forge of Empires\n");
        }
    }
}
```

### 6.2.2. Resource.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MiddleAges
{
    class Resource
    {
        public Resource(int _gold, int _wood, int _stone, int _iron, int
_food)
        {
            GoldQuantity = _gold;
            WoodQuantity = _wood;
            StoneQuantity = _stone;
            IronQuantity = _iron;
            FoodQuantity = _food;
        }

        //#####################################################
        // MEMBER FUNCTIONS
        //#####################################################
        // Remove
        public void RemoveGold(int quantity)
        {
            _goldQuantity -= quantity;
        }
        public void RemoveWood(int quantity)
        {
            _woodQuantity -= quantity;
        }
        public void RemoveStone(int quantity)
        {
            _stoneQuantity -= quantity;
        }
        public void RemoveIron(int quantity)
        {
            _ironQuantity -= quantity;
        }
        public void RemoveFood(int quantity)
        {
            _foodQuantity -= quantity;
        }

        // Add
        public void AddGold(int quantity)
        {
```

```
            _goldQuantity += quantity;
        }
        public void AddWood(int quantity)
        {
            _woodQuantity += quantity;
        }
        public void AddStone(int quantity)
        {
            _stoneQuantity += quantity;
        }
        public void AddIron(int quantity)
        {
            _ironQuantity += quantity;
        }
        public void AddFood(int quantity)
        {
            _foodQuantity += quantity;
        }

        //#################################################
        // SETTERS & GETTERS
        //#################################################

        public int GoldQuantity
        {
            get
            {
                return _goldQuantity;
            }
            set
            {
                _goldQuantity = value;
            }
        }

        public int WoodQuantity
        {
            get
            {
                return _woodQuantity;
            }
            set
            {
                _woodQuantity = value;
            }
        }

        public int StoneQuantity
        {
```

```
            get
            {
                return _stoneQuantity;
            }
            set
            {
                _stoneQuantity = value;
            }
        }

        public int IronQuantity
        {
            get
            {
                return _ironQuantity;
            }
            set
            {
                _ironQuantity = value;
            }
        }

        public int FoodQuantity
        {
            get
            {
                return _foodQuantity;
            }
            set
            {
                _foodQuantity = value;
            }
        }

        //#####################################################
        // MEMBER VARIABLES
        //#####################################################
        private int _goldQuantity;
        private int _woodQuantity;
        private int _stoneQuantity;
        private int _ironQuantity;
        private int _foodQuantity;
    }
}
```

### 6.2.3. Price.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MiddleAges
{
    class Price
    {
        //#####################################################
        // MEMBER FUNCTIONS
        //#####################################################
        public void IncreaseUpgradePrice(double gold, double wood, double
stone, double iron, double food)
        {
            _priceUpgradeGold += gold;
            _priceUpgradeWood += wood;
            _priceUpgradeStone += stone;
            _priceUpgradeIron += iron;
            _priceUpgradeFood += food;
        }

        public void IncreaseBuyPrice(double gold, double wood, double stone,
double iron, double food)
        {
            _priceBuyGold += gold;
            _priceBuyWood += wood;
            _priceBuyStone += stone;
            _priceBuyIron += iron;
            _priceBuyFood += food;
        }

        //#####################################################
        // GETTERS & SETTERS
        //#####################################################
        public double PriceBuyGold
        {
            get { return _priceBuyGold; }
            set { _priceBuyGold = value; }
        }
        public double PriceBuyWood
        {
            get { return _priceBuyWood; }
            set { _priceBuyWood = value; }
        }
        public double PriceBuyStone
        {
            get { return _priceBuyStone; }
            set { _priceBuyStone = value; }
```

```
        }
        public double PriceBuyIron
        {
            get { return _priceBuyIron; }
            set { _priceBuyIron = value; }
        }
        public double PriceBuyFood
        {
            get { return _priceBuyFood; }
            set { _priceBuyFood = value; }
        }

        public double PriceUpgradeGold
        {
            get { return _priceUpgradeGold; }
            set { _priceUpgradeGold = value; }
        }
        public double PriceUpgradeWood
        {
            get { return _priceUpgradeWood; }
            set { _priceUpgradeWood = value; }
        }
        public double PriceUpgradeStone
        {
            get { return _priceUpgradeStone; }
            set { _priceUpgradeStone = value; }
        }
        public double PriceUpgradeIron
        {
            get { return _priceUpgradeIron; }
            set { _priceUpgradeIron = value; }
        }
        public double PriceUpgradeFood
        {
            get { return _priceUpgradeFood; }
            set { _priceUpgradeFood = value; }
        }

        //#####################################################
        // MEMBER VARIABLES
        //#####################################################
        private double _priceBuyGold;
        private double _priceBuyWood;
        private double _priceBuyStone;
        private double _priceBuyIron;
        private double _priceBuyFood;

        private double _priceUpgradeGold;
        private double _priceUpgradeWood;
```

```
        private double _priceUpgradeStone;
        private double _priceUpgradeIron;
        private double _priceUpgradeFood;
    }
}
```

### 6.2.4. Building.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MiddleAges
{
    class Building : Price
    {
        //#####################################################
        // CONSTRUCTOR
        //#####################################################
        public Building() { }

        public Building(int _gold, int _wood, int _stone, double _dropRate =
1.0, int _level = 1)
        {
            PriceUpgradeGold = _gold;
            PriceUpgradeWood = _wood;
            PriceUpgradeStone = _stone;
            BuildingDropRate = _dropRate;
            BuildingLevel = _level;
        }

        //#####################################################
        // MEMBER FUNCTIONS
        //#####################################################
        public void IncreaseDropRate(double _value)
        {
            _buildingDropRate *= _value;
        }

        public void Upgrade(int _gold, int _wood, int _stone)
        {
            _buildingLevel += 1;
            switch (_buildingLevel)
            {
                case 1:
                    IncreaseUpgradePrice(_gold, _wood, _stone, 0, 0);
```

```
                    IncreaseDropRate(1);
                    break;
                case 2:
                    IncreaseUpgradePrice(1.1 * _gold, 1.1 * _wood, 1.1 *
_stone, 0, 0);
                    IncreaseDropRate(1.1);
                    break;
                case 3:
                    IncreaseUpgradePrice(1.2 * _gold, 1.2 * _wood, 1.2 *
_stone, 0, 0);
                    IncreaseDropRate(1.2);
                    break;
                case 4:
                    IncreaseUpgradePrice(1.5 * _gold, 1.5 * _wood, 1.5 *
_stone, 0, 0);
                    IncreaseDropRate(1.3);
                    break;
                case 5:
                    IncreaseUpgradePrice(1.8 * _gold, 1.8 * _wood, 1.8 *
_stone, 0, 0);
                    IncreaseDropRate(1.4);
                    break;
                default:
                    IncreaseUpgradePrice(0, 0, 0, 0, 0);
                    IncreaseDropRate(1);
                    break;
            }
        }


        //#####################################################
        // GETTERS & SETTERS
        //#####################################################
        public double BuildingDropRate
        {
            get { return _buildingDropRate; }
            set { _buildingDropRate = value; }
        }

        public int BuildingLevel
        {
            get { return _buildingLevel; }
            set { _buildingLevel = value; }
        }

        //#####################################################
        // MEMBER VARIABLES
        //#####################################################
        protected double _buildingDropRate;
```

```
        protected int _buildingLevel;
    }
}
```

### 6.2.5. Army.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MiddleAges
{
    class Army : Price
    {
        //#####################################################
        // CONSTRUCTORS
        //#####################################################

        //#####################################################
        // MEMBER FUNCTIONS
        //#####################################################
        public void AddQuantity(int value)
        {
            _quantity += value;
        }

        public void RemoveQuantity(int value)
        {
            _quantity -= value;
        }

        public void AddPower(int value)
        {
            _power += value;
        }

        public void RemovePower(int value)
        {
            _power -= value;
        }

        public void MultiplyPower(double value)
        {
            _power *= value;
        }
```

```
        //#################################################
        // GETTERS & SETTERS
        //#################################################
        public double Power
        {
            get { return _power;  }
            set { _power = value; }
        }

        public int Quantity
        {
            get { return _quantity; }
            set { _quantity = value; }
        }

        public int Level
        {
            get { return _level; }
            set { _level = value; }
        }

        //#################################################
        // MEMBER VARIABLES
        //#################################################
        protected double _power;
        protected int _quantity;
        protected int _level;

    }
}
```

### 6.2.6. MyArmy.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MiddleAges
{
    class MyArmy : Army
    {
        //#################################################
        // CONSTRUCTORS
        //#################################################
        public MyArmy() { }
```

```
        public MyArmy(int limit)
        {
            ArmyLimit = limit;
        }

        //#################################################
        // MEMBER FUNCTIONS
        //#################################################
        public void AddArmyLimit(int value)
        {
            _armyLimit += value;
        }

        public void Buy(int value, int quantity, int gold, int iron, int food)
        {
            AddQuantity(quantity);
            AddPower(value);
            IncreaseBuyPrice(gold, 0, 0, iron, food);
        }

        public void Upgrade(double powerRate, int gold, int food, int iron)
        {
            _level += 1;

            MultiplyPower(powerRate);

            switch (_level)
            {
                case 1:
                    IncreaseUpgradePrice(gold, 0, 0, iron, food);
                    break;
                case 2:
                    IncreaseUpgradePrice(1.1 * gold, 0, 0, 1.1 * iron, 1.1 *
food);
                    break;
                case 3:
                    IncreaseUpgradePrice(1.2 * gold, 0, 0, 1.2 * iron, 1.2 *
food);
                    break;
                case 4:
                    IncreaseUpgradePrice(1.3 * gold, 0, 0, 1.3 * iron, 1.3 *
food);
                    break;
                case 5:
                    IncreaseUpgradePrice(1.4 * gold, 0, 0, 1.4 * iron, 1.4 *
food);
                    break;
                default:
                    IncreaseUpgradePrice(0, 0, 0, 0, 0);
```

```
                    break;
            }
        }

        //#####################################################
        // GETTERS & SETTERS
        //#####################################################
        public int ArmyLimit
        {
            get { return _armyLimit; }
            set { _armyLimit = value; }
        }

        //#####################################################
        // MEMBER VARIABLES
        //#####################################################
        private int _armyLimit;
    }
}
```

### 6.2.7. Enemy.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MiddleAges
{
    class Enemy : Army
    {
        //#####################################################
        // CONSTRUCTORS
        //#####################################################
        public Enemy(double _enemyPower, int _rewardGold, int _rewardWood, int
_rewardStone, int _rewardIron, int _rewardFood)
        {
            Power = _enemyPower;
            RewardGold = _rewardGold;
            RewardWood = _rewardWood;
            RewardStone = _rewardStone;
            RewardIron = _rewardIron;
            RewardFood = _rewardFood;
        }

        //#####################################################
        // MEMBER FUNCTIONS
```

```
        //#################################################

        //#################################################
        // GETTERS & SETTERS
        //#################################################
        public int RewardGold
        {
            get { return _rewardGold; }
            set { _rewardGold = value; }
        }

        public int RewardWood
        {
            get { return _rewardWood; }
            set { _rewardWood = value; }
        }

        public int RewardStone
        {
            get { return _rewardStone; }
            set { _rewardStone = value; }
        }

        public int RewardIron
        {
            get { return _rewardIron; }
            set { _rewardIron = value; }
        }

        public int RewardFood
        {
            get { return _rewardFood; }
            set { _rewardFood = value; }
        }

        //#################################################
        // MEMBER VARIABLES
        //#################################################
        private int _rewardGold;
        private int _rewardWood;
        private int _rewardStone;
        private int _rewardIron;
        private int _rewardFood;
    }
}
```

# 7. Wnioski

## 7.1. C++ i C#

Tworząc grę w języku C++ korzystałem ze zbioru bibliotek Qt. Kod pisałem w Qt Editorze, a GUI modelowałem przy użyciu Qt Designera.

Wersję C# programowałem w środowisku Microsoft Visual Studio z użyciem frameworka WPF (Windows Presentation Foundation). GUI tworzyłem pisząc kod w języku XAML służącym do modelowania interfejsów użytkownika.

Kod napisany w języku C# w porównaniu z C++ jest znacznie krótszy, a sam proces programowania w owym języku był znacznie przyjemniejszy. Wpływ na to odczucie mają różnice w strukturze obu języków oraz frameeworków.

## 7.2. Polimorfizm

Zastosowanie polimorfizmu na większą skalę znacznie ułatwiłoby pisanie aplikacji. Proces programowania zająłby dużo mniej, kod byłby krótszy i czytelniejszy, a program skalowalny – np. dodanie nowej jednostki bądź budynku nie byłoby żadnym problemem.

## 7.3. Balans rozgrywki

Wartości początkowe surowców, koszty budynków, jednostek, nagrody za zwycięstwo w walce, częstotliwość poboru jedzenia przez żołnierzy i inne wartości nie są wielkościami zoptymalizowanymi pod zrównoważoną i przyjemną grę. Zbilansowanie rozgrywki jest czynnością złożoną i wymagającą wielu prób, a nie było celem projektu.