

Wojskowa Akademia Techniczna  
im. Jarosława Dąbrowskiego

Wydział Cybernetyki

Dokumentacja aplikacji Chess - Szachy  
wersja .1

Program wykonany na przedmiot: Metody programowania .NET

Autorzy: Tomasz Kita, Patryk Piasecki

Prowadzący: mgr inż. Kamil Małysz

## Opis aplikacji Chess – Szachy

Aplikacja Chess – Szachy to gra desktopowa, przeznaczona dla wszystkich miłośników szachów oraz osób chcących się nauczyć grać w tę klasyczną grę planszową. Aplikacja składa się z planszy i pionków - zarówno w kolorach białych (White) oraz czarnych (Black). W grze bierze udział dwóch graczy. Chess-Szachy to gra umożliwiająca przeprowadzenie standardowej partii szachów na komputerze. Zaletą gry jest możliwość wskazania na planszy wszystkich możliwych ruchów wykonywanych przez konkretnego pionka. Dzięki temu gra może służyć oprócz prowadzenia standardowych rozgrywek także do nauki zasad szachów.

## Wymagania techniczne:

Gra została stworzona z przeznaczeniem na komputer typu PC, wyposażone w monitor i myszkę oraz system operacyjny Windows 7. i wyższy.

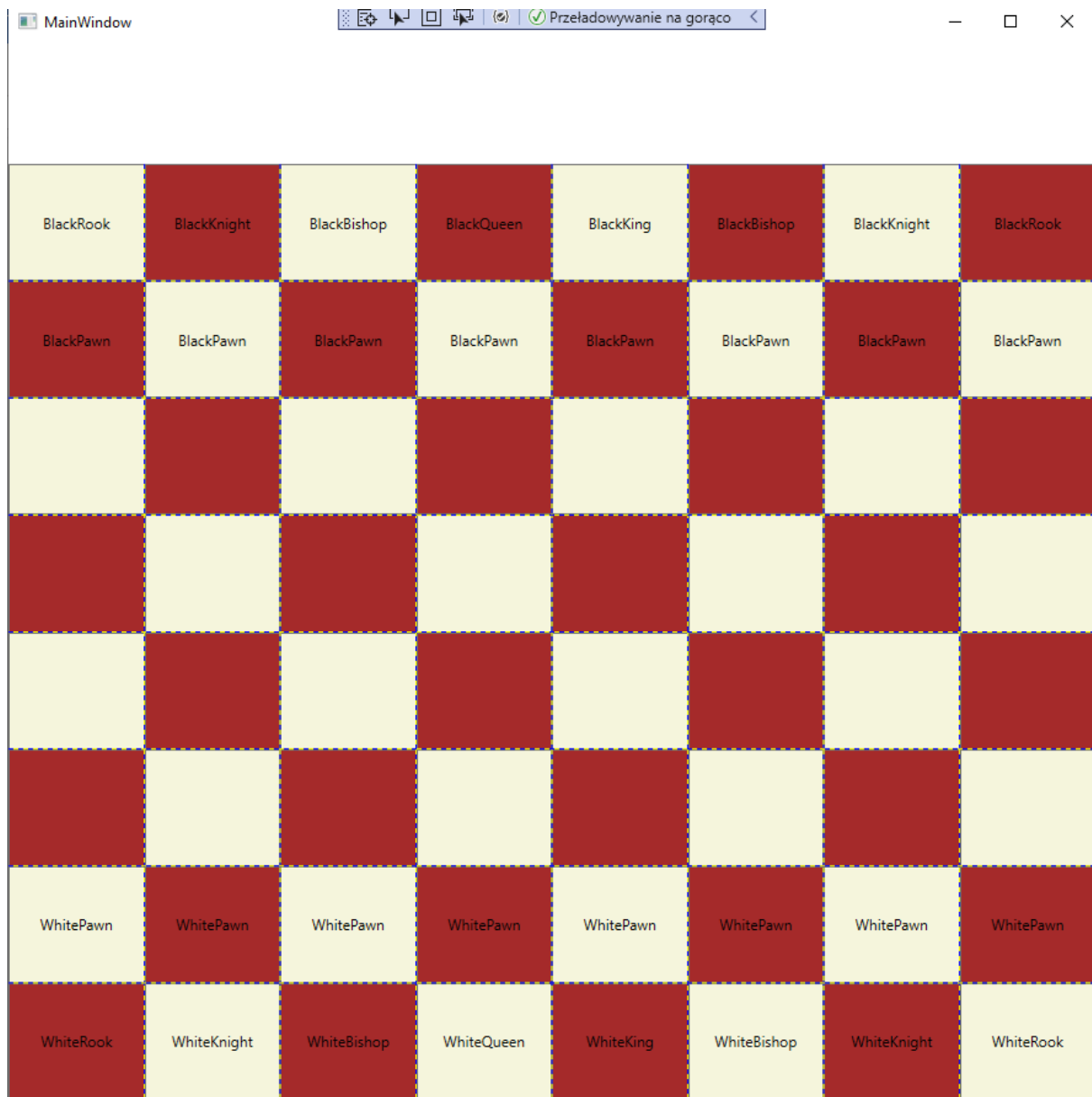
## Opis działania aplikacji:

Po uruchomieniu aplikacji pojawia się plansza z rozstawionymi pionkami. Gra nie wymaga żadnych dodatkowych ustawień, więc od razu można rozpocząć rozgrywkę.

W rozgrywce bierze udział dwóch graczy:

- biały (White), rozpoczynający,
- czarny (Black), wykonujący ruch po graczu białym.

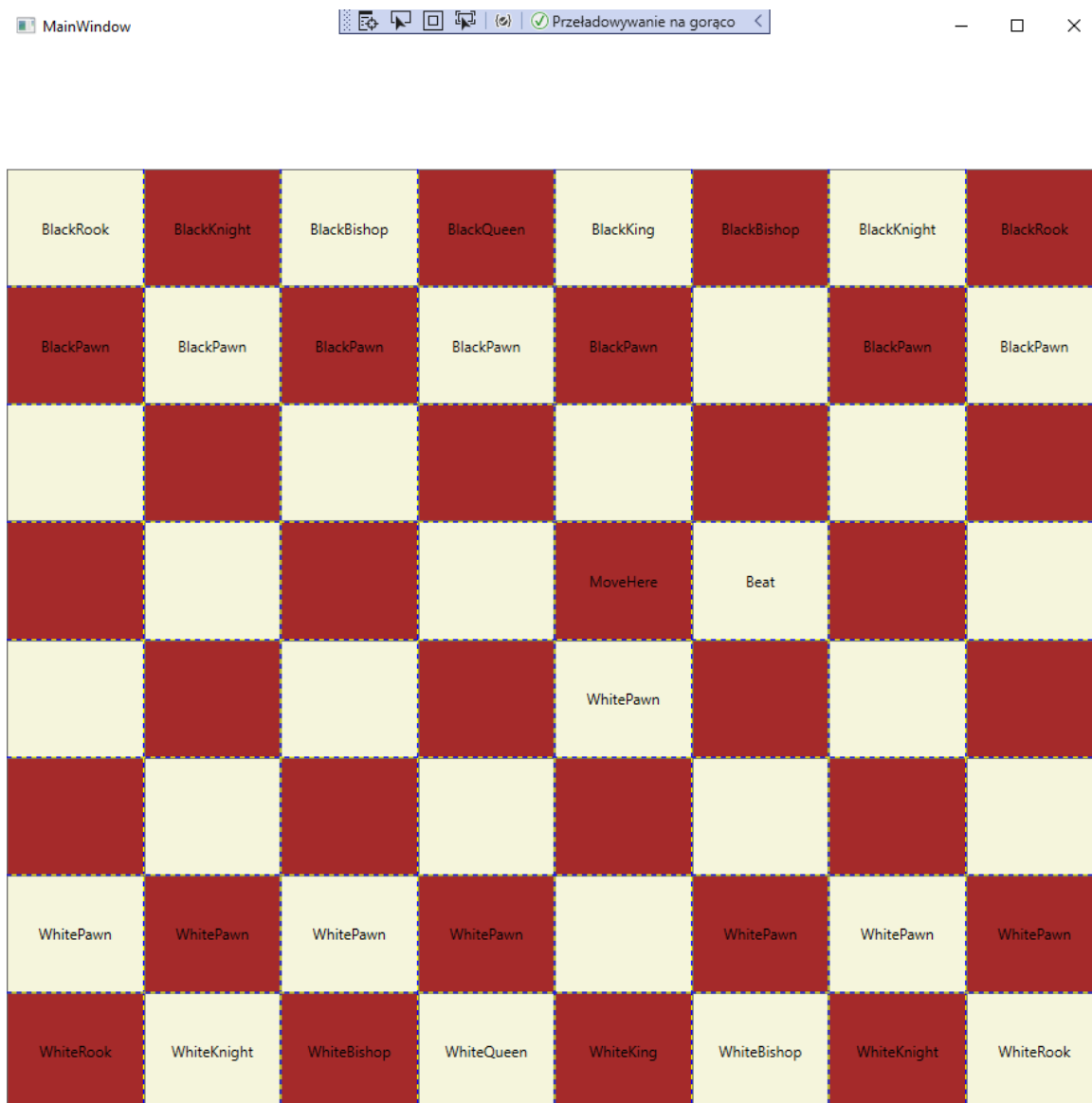
Poniżej znajduje się widok działającej aplikacji Chess-Szachy (początek gry, rozstawione pionki).



Ruchy pionkami wykonuje się klikając w konkretny pionek. Po kliknięciu, w niektórych polach na planszy pojawia się informacja o możliwych do wykonania ruchach. Ruchy dzielą się na:

- MoveHere – wskazanie na przesunięcie w pole gdzie nie stoi żaden pionek,
- Beat – powodujące, że mamy możliwość wykonania bicia pionka naszego przeciwnika.

Poniżej na szachownicy pokazane są możliwości wykonania ruchu i bicia przez pionka białego (White Pawn)













Gra posiada funkcjonalność sygnalizowania, że gracz kliknął nie w swój pionek. Wtedy pojawia się okienko z komunikatem, że ruch powinien wykonać inny gracz.

## Opis techniczny

Aplikacja została napisana w języku C# z wykorzystaniem frameworka .NET, aplikacja kliencka Windows Presentation Foundation.

Aplikacja składa się z następujących plików:

	App.config
	App.xaml
	App.xaml.cs
	Board.cs
	Cell.cs
	Chess1.csproj
	Chess1.sln
	MainWindow.xaml
	MainWindow.xaml.cs
	Player.cs

## MainWindow.xaml.cs

Za warstwę wyświetlania aplikacji Chess-Szachy na monitorze użytkownika odpowiada plik MainWindow.xaml. Jest tam umieszczony formularz, który jest głównym elementem aplikacji. Na nim został umieszczony element typu Grid o wymiarach 800px/800px. Grid posiada oznaczenie x:Name: grid. Grid został podzielony na 8 kolumn i 8 wierszy, które budują planszę.

W pliku MainWindow.xaml.cs zostały umieszczone funkcje:

- inicjalizująca wyświetlanie, InitializeComponent();
- wyświetlająca planszę startową, void showStartBoard();
- odpowiadająca za obsługę zdarzeń: kliknięć, void Grid\_Button\_Click(object sender, RoutedEventArgs e);
- wyświetlające (aktualizujące) planszę po ruchu innym niż pierwszy void showBoard();

Za szachownicę odpowiada Klasa Board. Instancją tej klasy jest myBoard, będący główną szachownicą, używaną w trakcie gry.

```
private static Board myBoard = new Board(int Size);
```

Size ustawiony jest standardowo na 8, co powoduje, że powstaje szachownica posiadająca 8 wierszy i 8 kolumn.

Za przyciski odpowiada tablica Button[,] btnGrid = new Button[myBoard.Size, myBoard.Size];

## Szczegółowy opis zmiennych:

Zmienna odpowiedzialna za szachownicę:

```
private static Board myBoard
```

Zmienna odpowiedzialna za przyciski (tablica dwuwymiarowa złożona z przycisków klasy Button:

```
private Button[,] btnGrid = new Button[myBoard.Size, myBoard.Size];
```

Zmienna przechowująca informację o poprzednio klikniętym polu na szachownicy:

```
private static Cell previousClickedCell;
```

Zmienna przechowująca informację o kolejności ruchów:

```
private bool WhiteTurn
```

Zmienna przechowująca informację czy ktoś wygrał:

```
private bool someoneWon
```

Zmienna przechowująca informację kto wygrał:

```
private string whoWon;
```

### Szczegółowy opis metod:

Metoda odpowiedzialna za wyświetlenie szachownicy na początku rozgrywki i uzupełnienie jej pionkami:

```
public void showStartBoard();
```

Metoda odpowiedzialna za wyświetlenie szachownicy w następujących po pierwszym ruchu.

```
public void showBoard();
```

### Klasa Board

Klasa Board odpowiada za zasady gry Chess-Szachy oraz konstrukcję Szachownicy (zmienna theGrid).

Tu znajdują się wszelkie algorytmy odpowiedzialne za ruchy pionków.

### Szczegółowy opis zmiennych:

Zmienna odpowiedzialna za wielkość szachownicy (8).

```
public int Size;
```

Zmienna tablicowa odpowiedzialna za przechowywanie wszystkich pól szachownicy.

```
public Cell[,] theGrid;
```

### Szczegółowy opis metod i funkcji:

Konstruktor tworzący szachownicę:

```
public Board(int s)
```

Metoda odpowiedzialna za wskazanie następnych możliwych wolnych ruchów. Przyjmuje pole pionka oraz nazwę wybranego pionka:

```
public void MarkNextLegalMoves(Cell currentCell, string chessPiece);
```

Pomocnicza funkcja wykorzystywana w metodzie MarkNextLegalMoves, niepozwalająca na "wyjście poza szachownicę":

```
public bool isSafe(int row, int col);
```

Przykład sprawdzania możliwych ruchów dla białej wieży:

```
case "WhiteRook":

    //checkLine(Cell currentCell, - 1, 0);
    for (int i = 1; i < Size; i++)
    {
        if (isSafe(y - i, x))
        {
            if (theGrid[y - i, x].PieceName.Contains("White"))
            {
                break;
            }
            else if (theGrid[y - i, x].PieceName.Contains("Black"))
            {
                theGrid[y - i, x].LegalNextMove = true;
                break;
            }
            else
                theGrid[y - i, x].LegalNextMove = true;
        }
        else break;
    }
    for (int i = 1; i < Size; i++)...

    for (int i = 1; i < Size; i++)...
    for (int i = 1; i < Size; i++)...

    break;
```

W powyższym przykładzie w pętli sprawdzane są kolejne pola na lewo od wieży:

Wpierw jest sprawdzana czy kolejne pole nie wychodzi poza planszę (funkcja isSafe), po czym sprawdza czy na polu nie znajduje białej figury (w takim wypadku przerywa, gdyż nie może go ani zbić czy przeskoczyć), następnie sprawdza czy nie znajduje się czarna figurą którą może zbić, ale nie może przeskoczyć (także przerywa pętlę), w przypadku wolnego pola zaznacza możliwy ruch i kontynuuje ruch.

Analogicznie sprawdzane są pozostałe możliwe kierunki.



## Klasa Cell

Klasa Cell odpowiada za pojedyncze pole w szachownicy (część składowa Board).

### Szczegółowy opis zmiennych:

Zmienna przechowująca numer wiersza pola (Cell):

```
public int RowNumber;
```

Zmienna przechowująca numer kolumny pola (Cell):

```
public int ColNumber;
```

Zmienna przechowująca informację czy dane pole (Cell) jest wolne czy zajęte:

```
public bool CurrentlyOccupied;
```

Zmienna przechowująca informację czy dane pole (Cell) może być zajęte przez wybrany przez gracza pionek:

```
public bool LegalNextMove;
```

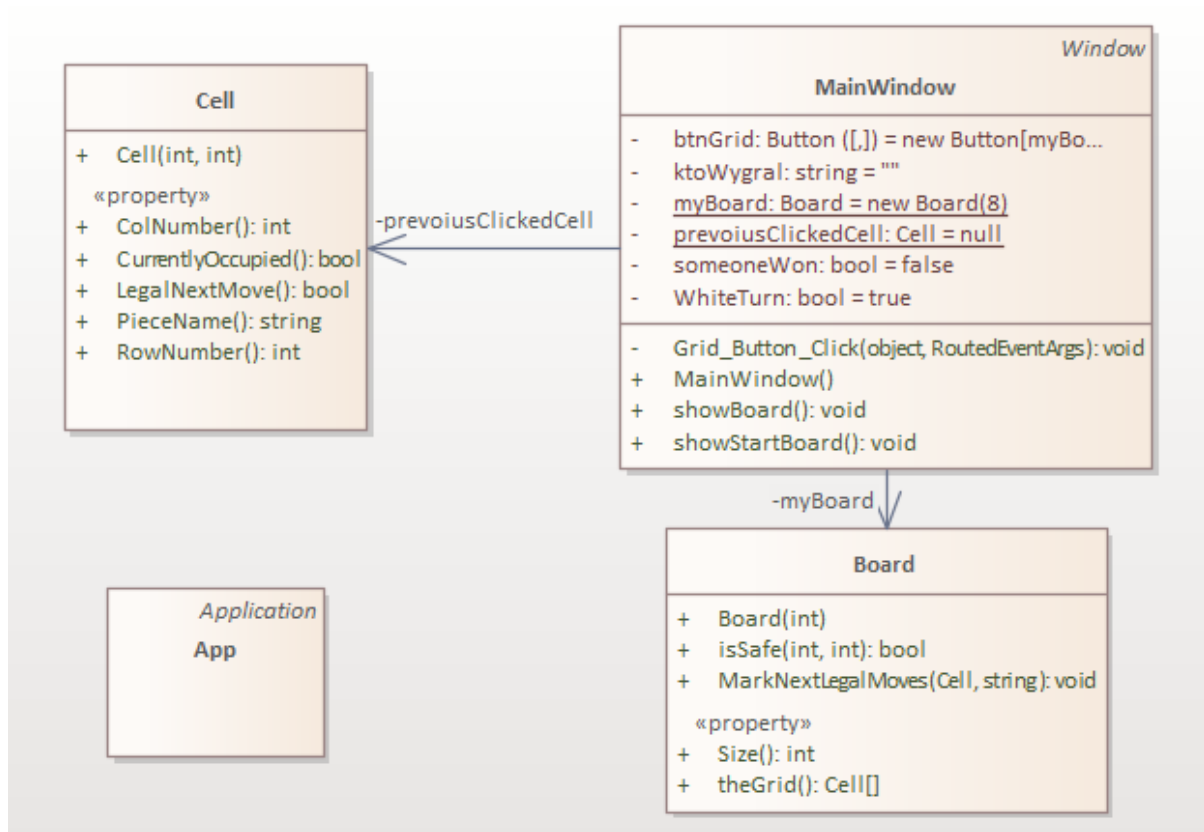
Zmienna przechowująca nazwę pionka:

```
public string PieceName
```

Zmienna niewykorzystywana w obecnej wersji Szachy-Chess;

```
public Player PieceColor;
```

Aplikacja Chess-Szachy – diagram klas:



W jaki sposób można poprawić kod:

Program nie odwzorowuje w pełni wszystkich opcji szachów. Wszystkie podstawowe ruchy figur są dostępne, ale brakuje opcji roszady dla króla z wieżą. Dodatkowo nie jest wprowadzona opcja szach (nie ostrzega o zagrożonym zbiciem królu) i szach-mat – gra się kończy jak zostanie zбитy król.

W miejsce napisów można wstawić obrazy symbolizujące poszczególne figury.

Kod sprawdzania w pętli linii możliwych ruchów jest kilkakrotnie powtarzana dla wieży, gońca i królowej - można ten kod zrefaktoryzować na wspólną funkcję skracając długość kodu.