Zachodniopomorski
Uniwersytet Technologiczny
w Szczecinie

**Wydział Inżynierii Mechanicznej i Mechatroniki**

Patryk Kołodziejski

Mateusz Kijak

**Projekt z przedmiotu: Inżynieria oprogramowania**

**Tytuł:**

# „ Gra w kółko i krzyżyk”

Kierunek studiów: Mechatronika

Prowadzący:

Data oddania:

13.06.2021

Szczecin 2021

## 1. Cel projektu:

Celem projektu jest zaprogramowanie aplikacji umożliwiającej zagranie w klasyczną grę „ kółko i krzyżyk”. Program aplikacji ma opierać się na wybranym wzorcu projektowym. Domyślnie aplikacja ma obsługiwać dwóch graczy kółko i krzyżyk oraz liczyć ich wygrane mecze. Aplikacja ma również zapewnić użytkownikowi możliwość gry jednoosobowej, „ z komputerem”

## 2. Założenia projektu:

a) Aplikacja ma opierać się o wybrany wzorzec projektowy
b) Użytkownik powinien wiedzieć ile razy wygrał, przegrał oraz czyja jest aktualnie kolej gry.
c) Użytkownik powinien mieć możliwość zresetowania gry
d) Poziom trudności gry z komputerem powinien zapewniać możliwość wygranej i przegranej. Odrzucamy algorytm, w którym użytkownik nie ma możliwości wygrania z komputerem.
e) Użytkowanie aplikacji powinno być proste, intuicyjne i przyjemne dla użytkownika.

## 3. Dobrany wzorzec projektowy do aplikacji:

W naszym projekcie został wykorzystany wzorzec projektowy MVC ( Model, View, Controler). Wzorzec projektowy zakłada podział naszej aplikacji na podzielenie kodu na trzy różne odpowiedzialne funkcje. Literka V, reprezentująca widok, jawnie sugeruje, że tego rodzaju wzorzec jest skierowany stricte do wdrożeń wykorzystujących GUI jako integralny element tworzonego rozwiązania.

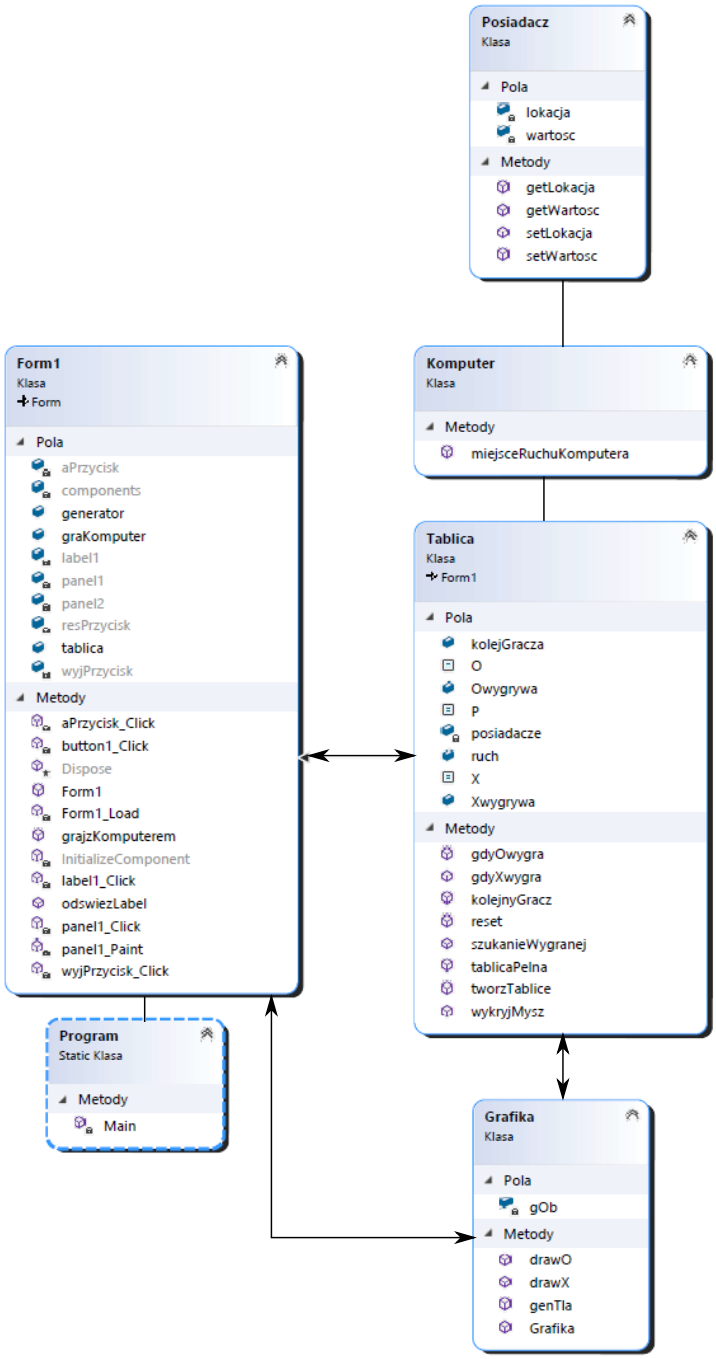
Kontroler to w pewnym sensie miejsce, od którego wszystko się zaczyna we wzorcu MVC (mimo że kryje się pod ostatnią literą nazwy).

W normalnej aplikacji to właśnie kontroler jest swoistą pierwszą linią w aplikacji:

- Wypełnia model.
- Przekazuje model do widoku.
- Reaguje na interakcje użytkownika stymulowane z poziomu widoku.

Dane przez kontroler powinny trafić do modelu. Model przeważnie jest więc klasą, która przechowuje dane oraz je przetwarza przy pośrednictwie kontrolera. Model jest przekazywany przez kontrolera do widoku, co umożliwia obustronną komunikację między tymi elementami.

# Diagram klas:



Kontrolerem w danym programie są następujące klasy:

- Form1
- Program

Modelem są:

- Posiadacz
- Komputer
- Tablica

Widokiem jest klasa Grafika.

Jest to podstawowy podział w wzorcu projektowym MVC. Kontroler ma za zadanie przetwarzać dane, w Form1 występują metody takie jak:

- aPrzycisk_Click- metoda ta obsługuje przycisk, naciśnięcie go powoduje wyświetlenie się komunikatu (przetwarza dane o naciśnięciu przycisku na wyświetlenie komunikatu)
- wyjPrzycisk_Click- metoda ta obsługuje przycisk, naciśnięcie go powoduje zamknięcie aplikacji

Zadaniem widoku jest przedstawienie grafiki aplikacji

- drawO, drawX- metody te rysują kółka i krzyżyki
- genTla- metoda ta odpowiedzialna jest za generowanie tła aplikacji

Zadaniem modelu jest przechowywanie informacji

- lokacja, wartosc- pola te przechowują informacje na temat tego, który gracz w którym miejscu kliknął
- kolejGracza- pole to przechowuje informacje o kolei gracza

## 4. Omówienie kodu

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Threading.Tasks;
 5  using System.Windows.Forms;
 6
 7  namespace KK_01
 8  {
 9      static class Program
10      {
11          /// <summary>
12          ///  The main entry point for the application.
13          /// </summary>
14          [STAThread]
15          static void Main()
16          {
17              Application.SetHighDpiMode(HighDpiMode.SystemAware);
18              Application.EnableVisualStyles();
19              Application.SetCompatibleTextRenderingDefault(false);
20              Application.Run(new Form1());
21          }
22      }
23  }
24
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.ComponentModel;
 4  using System.Data;
 5  using System.Drawing;
 6  using System.Linq;
 7  using System.Text;
 8  using System.Threading.Tasks;
 9  using System.Windows.Forms;
10
11  namespace KK_01
12  {
13      // KONTROLER ********************
14      public partial class Form1 : Form
15      {
16          public Grafika generator;
17          public Tablica tablica;
18          public static bool graKomputer = false;
19
20          // WYBRANIE GRY Z KOMPUTEREM
21          public static bool grajzKomputerem()
22          {
23              return graKomputer;
24          }
25          public Form1()
26          {
27              InitializeComponent();
28
29          }
30          // 1 panel gry odpowiedzialny za szachownice
31          private void panel1_Paint(object sender, PaintEventArgs e)
32          {
33              Graphics Przekaz = panel1.CreateGraphics();
34              generator = new Grafika(Przekaz);
35
36              tablica = new Tablica();
37              tablica.tworzTablice();
38
39              odswiezLabel();
40          }
41
42          private void panel1_Click(object sender, EventArgs e)
43          {
44              Point mysz = Cursor.Position;
45              mysz = panel1.PointToClient(mysz);
46              // określa gdzie zostało kliknięte
47              tablica.wykryjMysz(mysz);
48              // Odświeża Label w panelu
49              odswiezLabel();
50          }
51
52          // RESET gry
53          private void button1_Click(object sender, EventArgs e)
```

```
54          {
55              // Resetuje tablice
56              tablica.reset();
57              Grafika.genTla();
58          }
59
60          private void label1_Click(object sender, EventArgs e)
61          {
62
63          }
64
65          // PRZYCIŚNIĘCIE PRZYCISKU WIĘCEJ
66          private void aPrzycisk_Click(object sender, EventArgs e)
67          {
68              MessageBox.Show(" Projekt wykonany na zajęcia inżynierii
                oprogramowania MT 2020/2021");
69          }
70          // PRZYCISNIECIE PRZYCISKU WYJSCIE
71          private void wyjPrzycisk_Click(object sender, EventArgs e)
72          {
73              Application.Exit();
74          }
75          // ODŚWIEŻANIE LABEL 1 W OKNIE GRACZA
76          public void odswiezLabel()
77          {
78              String nowyTekst = " To kolej ";
79              if (tablica.kolejnyGracz() == Tablica.X)
80              {
81                  nowyTekst += "X";
82              }
83              else
84              {
85                  nowyTekst += "O";
86              }
87              nowyTekst += "  \n";
88              nowyTekst += " X wygrało " + tablica.gdyXwygra() + " razy \n O
                wygrało " + tablica.gdyOwygra() + " razy \n ";
89              label1.Text = nowyTekst;
90          }
91          // OKNO WYBORU GRY Z KOMPUTEREM LUB GRY WE DWOJE
92          private void Form1_Load(object sender, EventArgs e)
93          {
94              if (MessageBox.Show(" Czy chcesz grać z komputerem w", "Kółko i
                Krzyżyk", MessageBoxButtons.YesNo) == DialogResult.Yes )
95              {
96                  graKomputer = true;
97              }
98          }
99      }
100 }
101
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Text;
 4  using System.Drawing;
 5
 6  namespace KK_01
 7  {
 8      class Komputer
 9      {
10          // PODRZĘDNA KLASA MODELU
11          public static Posiadacz miejsceRuchuKomputera(Posiadacz[,] tablica)
12          {
13              Posiadacz ruch = new Posiadacz();
14              ruch.setWartosc(Tablica.O);
15
16              // sprawdza czy środek gry jest pusty
17              if (tablica[1, 1].getWartosc() == Tablica.P)
18              {
19                  ruch.setLokacja(new Point(1, 1));
20                  return ruch;
21              }
22              // sprawdzanie pionowo czy mamy możliwośc wygrania  - algorytm
                  ofensywy
                  komputera----------------------------------------------------
                  ------------------------------------------------
23
24              for (int x = 0; x <3; x++)
25              {
26                  // sprwdzamy czy występują w kolumnie w lokacjach 0,0 i 0,1
                      O
27                  if (tablica[x, 0].getWartosc() == Tablica.O && tablica[x,
                      1].getWartosc() == Tablica.O && tablica[x, 2].getWartosc()
                       == Tablica.P)
28                  {
29                      ruch.setLokacja(new Point(x, 2));
30                      return ruch;
31                  }
32              }
33              for (int x = 0; x < 3; x++)
34              {
35                  // sprwdzamy czy występują w kolumnie w lokacjach 0,1 i 0,2
                      O
36                  if (tablica[x, 0].getWartosc() == Tablica.P && tablica[x,
                      1].getWartosc() == Tablica.O && tablica[x, 2].getWartosc()
                       == Tablica.O)
37                  {
38                      ruch.setLokacja(new Point(x, 0));
39                      return ruch;
40                  }
41              }
42              for (int x = 0; x < 3; x++)
43              {
44
```

```
45                  if (tablica[x, 0].getWartosc() == Tablica.O && tablica[x,
                      1].getWartosc() == Tablica.P && tablica[x, 2].getWartosc()
                       == Tablica.O)
46                  {
47                      ruch.setLokacja(new Point(x, 1));
48                      return ruch;
49                  }
50              }
51          // Sprawdzanie poziomo możliwości wygrania
52              for (int y = 0; y < 3; y++)
53              {
54
55                  if (tablica[0, y].getWartosc() == Tablica.O && tablica[1,
                      y].getWartosc() == Tablica.O && tablica[2, y].getWartosc()
                       == Tablica.P)
56                  {
57                      ruch.setLokacja(new Point(2, y));
58                      return ruch;
59                  }
60              }
61              for (int y = 0; y < 3; y++)
62              {
63
64                  if (tablica[0, y].getWartosc() == Tablica.P && tablica[1,
                      y].getWartosc() == Tablica.O && tablica[2, y].getWartosc()
                       == Tablica.O)
65                  {
66                      ruch.setLokacja(new Point(0, y));
67                      return ruch;
68                  }
69              }
70              for (int y = 0; y < 3; y++)
71              {
72
73                  if (tablica[0, y].getWartosc() == Tablica.P && tablica[1,
                      y].getWartosc() == Tablica.P  && tablica[2, y].getWartosc
                      () == Tablica.O)
74                  {
75                      ruch.setLokacja(new Point(1, y));
76                      return ruch;
77                  }
78              }
79          // Sprawdzanie wygranej na przekątnej
80              if (tablica[0,0].getWartosc() == Tablica.O && tablica
                  [1,1].getWartosc() == Tablica.O && tablica[2,2].getWartosc()
                  == Tablica.P)
81              {
82                  ruch.setLokacja(new Point(2, 2));
83                  return ruch;
84              }
85              else if (tablica[2, 0].getWartosc() == Tablica.O && tablica[1,
                  1].getWartosc() == Tablica.O && tablica[0, 2].getWartosc() ==
                  Tablica.P)
```

```
86              {
87                  ruch.setLokacja(new Point(0, 2));
88                  return ruch;
89              }
90              //
91              if (tablica[0, 2].getWartosc() == Tablica.O && tablica[1,
                  1].getWartosc() == Tablica.O && tablica[2, 0].getWartosc() ==
                  Tablica.P)
92              {
93                  ruch.setLokacja(new Point(0, 2));
94                  return ruch;
95              }
96              else if (tablica[2, 2].getWartosc() == Tablica.O && tablica[1,
                  1].getWartosc() == Tablica.O && tablica[0, 0].getWartosc() ==
                  Tablica.P)
97              {
98                  ruch.setLokacja(new Point(0, 0));
99                  return ruch;
100             }
101             if (tablica[0, 0].getWartosc() == Tablica.O && tablica[1,
                  1].getWartosc() == Tablica.P && tablica[2, 0].getWartosc() ==
                  Tablica.O)
102             {
103                 ruch.setLokacja(new Point(1, 1));
104                 return ruch;
105             }
106             else if (tablica[0, 2].getWartosc() == Tablica.O && tablica[1,
                  1].getWartosc() == Tablica.P && tablica[2, 0].getWartosc() ==
                  Tablica.O)
107             {
108                 ruch.setLokacja(new Point(1, 1));
109                 return ruch;
110             }
111             // koniec algorytmu odpowiadającego za
                  ofensywe----------------------------------------------------
                  ----------------------------------------------------------------
                  ---------------
112
113             // Defensywa Komputera
                  ----------------------------------------------------------------
                  ----------------------------------------------------------------
                  --------------------------------
114
115
116             for (int x = 0; x < 3; x++)
117             {
118                 // sprwdzamy czy występują w kolumnie w lokacjach X,0 i X,1
                      X
119                 if (tablica[x, 0].getWartosc() == Tablica.X && tablica[x,
                      1].getWartosc() == Tablica.X && tablica[x, 2].getWartosc()
                       == Tablica.P)
120                 {
121                     ruch.setLokacja(new Point(x, 2));
```

```
122                     return ruch;
123                 }
124             }
125             for (int x = 0; x < 3; x++)
126             {
127                 // sprwdzamy czy występują w kolumnie w lokacjach X,1 i X,2 X
128                 if (tablica[x, 0].getWartosc() == Tablica.P && tablica[x, 1].getWartosc() == Tablica.X && tablica[x, 2].getWartosc() == Tablica.X)
129                 {
130                     ruch.setLokacja(new Point(x, 0));
131                     return ruch;
132                 }
133             }
134             for (int x = 0; x < 3; x++)
135             {
136                 // sprwdzamy czy występują w kolumnie w lokacjach X,0 i X,2 X
137                 if (tablica[x, 0].getWartosc() == Tablica.X && tablica[x, 1].getWartosc() == Tablica.P && tablica[x, 2].getWartosc() == Tablica.X)
138                 {
139                     ruch.setLokacja(new Point(x, 1));
140                     return ruch;
141                 }
142             }
143             // Sprawdzanie poziomo możliwości przegrania
144             for (int y = 0; y < 3; y++)
145             {
146                 // sprwdzamy czy występują w kolumnie w lokacjach 0,y i 1,y X
147                 if (tablica[0, y].getWartosc() == Tablica.X && tablica[1, y].getWartosc() == Tablica.X && tablica[2, y].getWartosc() == Tablica.P)
148                 {
149                     ruch.setLokacja(new Point(2, y));
150                     return ruch;
151                 }
152             }
153             for (int y = 0; y < 3; y++)
154             {
155                 // sprwdzamy czy występują w kolumnie w lokacjach 1,y i 2,y X
156                 if (tablica[0, y].getWartosc() == Tablica.P && tablica[1, y].getWartosc() == Tablica.X && tablica[2, y].getWartosc() == Tablica.X)
157                 {
158                     ruch.setLokacja(new Point(0, y));
159                     return ruch;
160                 }
161             }
162             for (int y = 0; y < 3; y++)
```

```
163             {
164                 // sprwdzamy czy występują w kolumnie w lokacjach 0,y i 2,y X
165                 if (tablica[0, y].getWartosc() == Tablica.X && tablica[1, y].getWartosc() == Tablica.P && tablica[2, y].getWartosc() == Tablica.X)
166                 {
167                     ruch.setLokacja(new Point(1, y));
168                     return ruch;
169                 }
170             }
171             // Sprawdzanie przegranej na przekątnej
172             if (tablica[0, 0].getWartosc() == Tablica.X && tablica[1, 1].getWartosc() == Tablica.X && tablica[2, 2].getWartosc() == Tablica.P)
173             {
174                 ruch.setLokacja(new Point(2, 2));
175                 return ruch;
176             }
177             else if (tablica[2, 0].getWartosc() == Tablica.X && tablica[1, 1].getWartosc() == Tablica.X && tablica[0, 2].getWartosc() == Tablica.P)
178             {
179                 ruch.setLokacja(new Point(0, 2));
180                 return ruch;
181             }
182             //
183             if (tablica[0, 2].getWartosc() == Tablica.X && tablica[1, 1].getWartosc() == Tablica.X && tablica[2, 0].getWartosc() == Tablica.P)
184             {
185                 ruch.setLokacja(new Point(0, 2));
186                 return ruch;
187             }
188             else if (tablica[2, 2].getWartosc() == Tablica.X && tablica[1, 1].getWartosc() == Tablica.X && tablica[0, 0].getWartosc() == Tablica.P)
189             {
190                 ruch.setLokacja(new Point(0, 0));
191                 return ruch;
192             }
193             if (tablica[0, 0].getWartosc() == Tablica.X && tablica[1, 1].getWartosc() == Tablica.P && tablica[2, 0].getWartosc() == Tablica.X)
194             {
195                 ruch.setLokacja(new Point(1, 1));
196                 return ruch;
197             }
198             else if (tablica[0, 2].getWartosc() == Tablica.X && tablica[1, 1].getWartosc() == Tablica.P && tablica[2, 0].getWartosc() == Tablica.X)
199             {
200                 ruch.setLokacja(new Point(1, 1));
```

```
201                     return ruch;
202                 }
203                 //
204                 //
205                 //
206                 //
207                 // W przypadku nie osiągnięcia z żadnych ww. warunków ruszamy
                      gdziekolwiek
208
209                 List<Posiadacz> pusteMiejsca = new List<Posiadacz>();
210                 foreach (Posiadacz h in tablica)
211                 {
212                     if (h.getWartosc() == Tablica.P)
213                     {
214                         pusteMiejsca.Add(h);
215                     }
216                 }
217                 ruch.setLokacja(pusteMiejsca.ToArray()[0].getLokacja());
218                 return ruch;
219             }
220         }
221     }
222
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Text;
 4  using System.Windows.Forms;
 5  using System.Drawing;
 6  namespace KK_01
 7  {
 8      // MODEL ******************
 9      public class Tablica : Form1
10      {
11          // OKREŚLA KTÓRY RUCH
12          public int ruch = 0;
13          // OKREŚLA ILE WYGRAŁO RAZY O
14          public int Owygrywa = 0;
15          // OKREŚLA ILE WYGRAŁO RAZY X
16          public int Xwygrywa = 0;
17
18          private Posiadacz[,] posiadacze = new Posiadacz[3, 3];
19          // WARTOŚĆ SEKCJI Z  X
20          public const int X = 0;
21          // WARTOŚĆ SEKCJI Z  O
22          public const int O = 1;
23          // WARTOŚĆ SEKCJI Z  PUSTEJ
24          public const int P = 2;
25
26          public int kolejGracza = X;
27          public int kolejnyGracz()
28          {
29              return kolejGracza;
30          }
31          public int gdyOwygra()
32          {
33              return Owygrywa;
34          }
35          public int gdyXwygra()
36          {
37              return Xwygrywa;
38          }
39          // GENERUJEMY TABLICE [3,3] PUSTYCH LOKACJI
40          public void tworzTablice()
41              {
42              for (int i = 0; i < 3; i++)
43                  {
44                      for (int j = 0; j < 3; j++)
45                      {
46                      posiadacze[i, j] = new Posiadacz();
47                      posiadacze[i, j].setWartosc(P);
48                      posiadacze[i, j].setLokacja(new Point(i, j));
49                      }
50
51                  }
52              }
53          public void wykryjMysz(Point loc)
```

```
54          {
55              // Sprawdzamy czy kliknięcie jest w tablice gry i dokładnie w
                  którym sektorze kliknelismy
56              if (loc.Y <= 500)
57              {
58                  int x = 0;
59                  int y = 0;
60                  if (loc.X < 167)
61                  {
62                      x = 0;
63                  }
64                  else if (loc.X > 167 && loc.X < 334)
65                  {
66                      x = 1;
67                  }
68                  else if (loc.X > 334)
69                  {
70                      x = 2;
71                  }
72                  if (loc.Y < 167)
73                  {
74                      y = 0;
75                  }
76                  else if (loc.Y > 167 && loc.Y < 334)
77                  {
78                      y = 1;
79                  }
80                  else if (loc.Y > 334 && loc.Y < 500)
81                  {
82                      y = 2;
83                  }
84
85                  // SPRAWDZAMY CZY KOLEJ RUCHU X
86                  if (ruch % 2 ==0)
87                  {
88                      Grafika.drawX(new Point(x, y));
89                      posiadacze[x, y].setWartosc(X);
90                      if (szukanieWygranej())
91                      {
92                          MessageBox.Show("Wygrałeś gracz X");
93                          Xwygrywa++;
94                          reset();
95                          Grafika.genTla();
96
97                      }
98                      if (tablicaPelna())
99                      {
100                         reset();
101                         ruch++;
102
103                     }
104                     // SPRAWDZAMY CZY KOLEJ RUCHU KOMPUTERA
105                     if (Form1.grajzKomputerem() && !szukanieWygranej() && !
```

```
                    tablicaPelna())
106                  {
107                      Posiadacz ruchKomputera =
                    Komputer.miejsceRuchuKomputera(posiadacze);
108
109                      Grafika.drawO(ruchKomputera.getLokacja());
110                      posiadacze[ruchKomputera.getLokacja().X,
                    ruchKomputera.getLokacja().Y ].setWartosc(O);
111                      if (szukanieWygranej())
112                      {
113                          MessageBox.Show("Wygrał komputer");
114                          Owygrywa++;
115                          reset();
116                          Grafika.genTla();
117                      }
118                      ruch++;
119                      kolejGracza = X;
120
121                  }
122                  kolejGracza = O;
123              }
124              // W INNYM PRZYPADKU JEST TO KOLEJ GRACZA O
125              else
126              {
127                  Grafika.drawO(new Point(x, y));
128                  posiadacze[x, y].setWartosc(O);
129                  if (szukanieWygranej())
130                  {
131                      MessageBox.Show("Wygrałeś gracz O");
132                      Owygrywa++;
133                      reset();
134                      Grafika.genTla();
135                  }
136                  kolejGracza = X;
137              }
138              ruch++;
139
140              // pokazywanie orientacji kliknięcia oraz sektora
141              // MessageBox.Show(x.ToString() + "," + y.ToString() + "\n
                   \n" + loc.ToString());
142
143          }
144      }
145      // Sprawdzanie wygranej
146      public bool szukanieWygranej()
147      {
148          bool Wygrana = false;
149          // Sprawdzanie wygranych na liniach poziomych, pionowych oraz
               przekątnych
150          for (int x = 0; x < 3; x++)
151          {
152              if (posiadacze[x, 0].getWartosc() == X && posiadacze[x,
                    1].getWartosc() == X && posiadacze[x, 2].getWartosc() ==
```

```
                    X)
153                 {
154                     return true;
155                 }
156                 if (posiadacze[x, 0].getWartosc() == O && posiadacze[x,
                    1].getWartosc() == O && posiadacze[x, 2].getWartosc() ==
                    O)
157                 {
158                     return true;
159                 }
160                 switch (x)
161                 {
162                     case 0:
163                         if (posiadacze[x, 0].getWartosc() ==  X &&
                        posiadacze[x+1, 1].getWartosc() == X && posiadacze[x+2,
                        2].getWartosc() == X)
164                         {
165                             return true;
166                         }
167                         if (posiadacze[x, 0].getWartosc() == O && posiadacze
                        [x + 1, 1].getWartosc() == O && posiadacze[x + 2,
                        2].getWartosc() == O)
168                         {
169                             return true;
170                         }
171                         break;
172                     case 2:
173                         if (posiadacze[x, 0].getWartosc() == X && posiadacze
                        [x - 1, 1].getWartosc() == X && posiadacze[x - 2,
                        2].getWartosc() == X)
174                         {
175                             return true;
176                         }
177                         if (posiadacze[x, 0].getWartosc() == O && posiadacze
                        [x - 1, 1].getWartosc() == O && posiadacze[x - 2,
                        2].getWartosc() == O)
178                         {
179                             return true;
180                         }
181                         break;
182
183                 }
184             }
185             for (int y = 0; y < 3; y++)
186             {
187                 if (posiadacze[0, y].getWartosc() == X && posiadacze[1,
                    y].getWartosc() == X && posiadacze[2, y].getWartosc() ==
                    X)
188                 {
189                     return true;
190                 }
191                 if (posiadacze[0, y].getWartosc() == O && posiadacze[1,
                    y].getWartosc() == O && posiadacze[2, y].getWartosc() ==
```

```
                    O)
192                 {
193                     return true;
194                 }
195             }
196             return Wygrana;
197
198         }
199         // RESETOWANIE TABLICY
200         public void reset()
201         {
202             posiadacze = new Posiadacz[3, 3];
203             tworzTablice();
204         }
205         // SPRAWDZANIE CZY TABLICA NIE JEST ZAPEŁNIONA CAŁKOWICIE - W TAKIM
              PRZYPADKU NIE MAMY MOŻLIWOŚCI RUCHU
206         public bool tablicaPelna()
207         {
208             bool pelna = true;
209             foreach (Posiadacz h in posiadacze)
210             {
211                 if (h.getWartosc() == Tablica.P)
212                 {
213                     pelna = false;
214                 }
215             }
216             return pelna;
217         }
218     }
219 }
220
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Text;
 4  using System.Drawing;
 5  using System.Linq;
 6
 7
 8  namespace KK_01
 9  {
10      // PODRZĘDNA KLASA MODELU
11      public class Posiadacz
12      {
13          private Point lokacja;
14          private int wartosc = Tablica.P;
15          // USTAWIANIE LOKACJI
16          public void setLokacja(Point p)
17          {
18              lokacja = p;
19          }
20          // POBIERANIE LOKACJI
21          public Point getLokacja()
22          {
23              return lokacja;
24          }
25          // USTAWIANIE WARTSCI SEKTORA
26          public void setWartosc(int i)
27          {
28              wartosc = i;
29          }
30          // POBIERANIE WARTOSCI SEKTORA
31          public int getWartosc()
32          {
33              return wartosc;
34          }
35
36      }
37  }
38
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Text;
 4  using System.Drawing;
 5  using System.Windows.Forms;
 6
 7
 8  namespace KK_01
 9  {
10      // VIEW *************************
11      public class Grafika
12      {
13          private static Graphics gOb;
14          public Grafika(Graphics g)
15          {
16              gOb = g;
17              genTla();
18          }
19          // GENEROWANIE TŁA APLIKACJI
20          public static void genTla()
21          {
22              Brush tlo = new SolidBrush(Color.PaleGreen);
23              Pen linia = new Pen(Color.Black, 4);
24
25              gOb.FillRectangle(tlo, new Rectangle(0, 0, 500, 600));
26              // RYSOWANIE LINII OKREŚLAJACYCH SEKTORY
27              gOb.DrawLine(linia, new Point(167, 0), new Point(167, 500));
28              gOb.DrawLine(linia, new Point(334, 0), new Point(334, 500));
29              gOb.DrawLine(linia, new Point(0, 167), new Point(500, 167));
30              gOb.DrawLine(linia, new Point(0, 334), new Point(500, 334));
31              gOb.DrawLine(linia, new Point(0, 500), new Point(500, 500));
32
33          }
34          // Rysowanie kształtu  X
35          public static void drawX(Point loc)
36          {
37              Pen xPen = new Pen(Color.Red, 5);
38              int xValue = loc.X * 167;
39              int yValue = loc.Y * 167;
40
41              gOb.DrawLine(xPen, xValue + 20, yValue + 20,xValue + 147, yValue
                  + 147);
42              gOb.DrawLine(xPen, xValue + 147, yValue + 20, xValue + 20, yValue
                    + 147);
43          }
44          // Rysowanie kształtu O
45          public static void drawO(Point loc)
46          {
47              Pen oPen = new Pen(Color.Blue, 5);
48              int xValue = loc.X * 167;
49              int yValue = loc.Y * 167;
50
51              gOb.DrawEllipse(oPen, xValue + 10, yValue + 10, 147, 147);
```

```
52          }
53      }
54  }
55
```

## 6. Instrukcja obsługi aplikacji: Instrukcja obsługi aplikacji:

Po starcie aplikacji wyskakuje okno. Jeśli chcesz grać z komputerem naciśnij „Tak”, jeśli chcesz włączyć tryb dwuosobowy wciśnij „Nie”

Najazd kursorem myszy na danych obszar i naciśniecie lewego przycisku myszy powoduje narysowanie na nim kółka lub krzyżyka. Rodzaj narysowanego symbolu zależy od tego, którego gracza jest kolei.

Gdy chcesz zacząć grę od nowa naciśnij przycisk „Reset”

Gdy chcesz wyjść z gry naciśnij przycisk „Wyjście”

Gdy chcesz uzyskać więcej informacji o aplikacji naciśnij przycisk „Więcej”

## 7. Wnioski:

Wzorzec projektowy MVC ułatwia podział pracy. Składa się on z trzech głównych „folderów”. Każdy z nich może wykonywać inna osoba. Mateusz Kijak był odpowiedzialny za widok i kontroler , a Patryk Kołodziejski za model i ogólną logikę programu . Z pomocą tego wzorca w początkowej fazie pisania programu nie musieli współpracować. W późniejszej fazie dopiero w celu połączenia programu w całość.