**TYTUŁ PROJEKTU**

22. Gra w kółko i krzyżyk (bez AI)

**GRUPA PROJEKTOWA**

Marta Augustyniak K25

Michał Wylegała K25

**ZAŁOŻENIA PROJEKTU**

Kryteria oceny:

1. Zarys dokumentacji:
   1. w przypadku projektu zawierającego tylko i wyłącznie API: 0 – 20 pkt,
   2. w przypadku pozostałych projektów: 0 – 10 pkt,
2. 2. Wykorzystanie poznanych mechanizmów obiektowych:
   1. w przypadku projektu zawierającego tylko i wyłącznie API: 0 – 10 pkt,
   2. w przypadku pozostałych projektów: 0 – 20 pkt,
3. Stosowanie konwencji nazewniczych w programowaniu i formatowania kodu typowego dla wybranej technologii: 0 – 5 pkt,
4. Obsługa wyjątków: 0 – 5 pkt,
5. Wykorzystanie poznanych mechanizmów sieciowych i/lub wątków: 0 – 10 pkt,
6. Przekazanie projektu za pomocą systemu kontroli wersji Git: 0 – 5 pkt;

W przypadku przekroczenia 50 pkt, np. wskutek osiągnięcia maksymalnej punktacji za projekt oraz przekazania projektu za pomocą systemu kontroli wersji Git, wynik końcowy zostanie zaokrąglony w dół, do maksymalnie 50 pkt.

W ramach powyższych kryteriów punktowane będą następujące elementy.

1. Zarys dokumentacji:
   1. krótki opis dostępnych funkcjonalności aplikacji,
   2. przedstawienie uproszczonego diagramu klas i zależności między nimi,
   3. (w przypadku projektu zawierającego tylko i wyłącznie API) opis wyeksponowanych metod, wystarczający do wdrożenia, z poziomu zewnętrznego systemu, integracji z utworzonym API na ich podstawie;
2. Wykorzystanie poznanych mechanizmów obiektowych:
   1. własne klasy z odpowiednio ustawionym poziomem dostępu do jej składowych,
   2. mechanizm dziedziczenia klas,
   3. zastosowanie metod lub metody statycznej,
   4. (nie dotyczy projektów zawierających tylko i wyłącznie API) zastosowanie konstruktora w klasie macierzystej i potomnej,
   5. (nie dotyczy projektów zawierających tylko i wyłącznie API) dowolny mechanizm polimorficzny,
   6. (nie dotyczy projektów zawierających tylko i wyłącznie API) zastosowanie klasy abstrakcyjnej lub interfejsu;
3. Stosowanie konwencji nazewniczych w programowaniu i formatowania kodu typowego dla wybranej technologii:
   1. konwencja nazewnictwa w ramach klas (klasa, zmienne, metody, właściwości) i interfejsów,
   2. (w przypadku projektów z obsługą zdarzeń) konwencja nazewnictwa w ramach zdarzeń,
   3. formatowanie kodu – wcięcia i odstępy w stosownych miejscach;
4. Obsługa wyjątków:
   1. przechwycenie i obsługa co najmniej jednego wyjątku w ramach aplikacji;
5. Wykorzystanie poznanych mechanizmów sieciowych i/lub wątków (wystarczy wybrać jedno z poniższych):
   1. wykorzystanie gniazd,
   2. wykorzystanie technologii .NET Remoting lub ekwiwalentnej,
   3. stworzenie metody lub metod API w technologii WebAPI lub ekwiwalentnej,
   4. wykorzystanie osobnego wątku lub wątków w aplikacji;
6. Przekazanie projektu za pomocą systemu kontroli wersji Git:
   1. cóż… przekazanie Prowadzącemu projektu z wykorzystaniem systemu kontroli wersji Git

**ROZWIĄZANIE PROJEKTU**

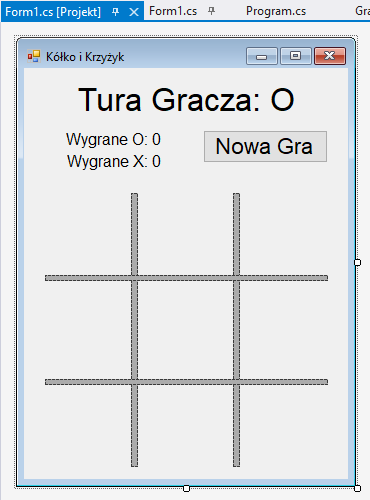
**Zarys dokumentacji:**

* **krótki opis dostępnych funkcjonalności aplikacji**

Prace nad wdrożeniem aplikacji rozpoczęliśmy od napisania od stworzenia nowego projektu Windows Forms, dodania wymaganych komponentów i zdefiniowania ich wyglądu. Następnie przeszliśmy do kodowania zdarzeń i implementacji logiki gry bez tworzenia dodatkowych klas.

Pierwszą wersję projektu załączamy z dopiskiem „\_beta” do naszego projektu.

Interfejs użytkownika dla gry Kółko i Krzyżyk zdefiniowaliśmy za pomocą Windows Forms i wygląda on tak:



Podstawową funkcjonalnością aplikacji jest umożliwienie rozgrywki w grę Kółko i Krzyżyk.

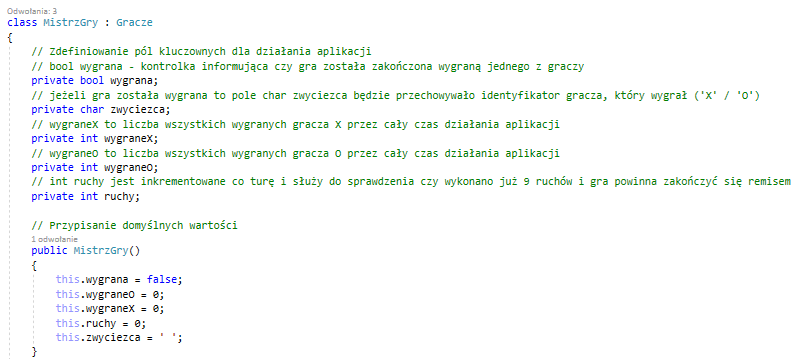
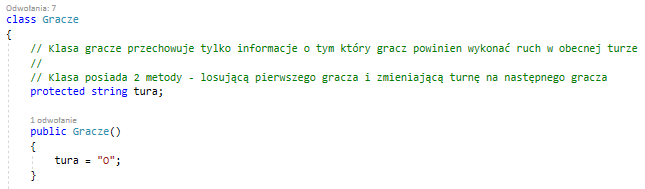
Gra pozwala na statyczne wybranie gracza rozpoczynającego rozgrywkę, lub wylosowanie pierwszego gracza (domyślna opcja). Istotnym dla funkcjonalności jest MistrzGry(), który czuwa nad przebiegiem gry sprawdzając czy gracz wykonał zwycięski ruch, monitorujący remis i umożliwiający rozpoczęcie nowej gry. Aplikacja pozwala też śledzić sumaryczną liczbę gier wygranych przez każdego z graczy. Podczas uruchomienia aplikacji otwierają się 2 okna gry, każde obsługiwane przez osobny, równorzędny wątek programu.

* **przedstawienie uproszczonego diagramu klas i zależności między nimi,**

**Wykorzystanie poznanych mechanizmów obiektowych:**

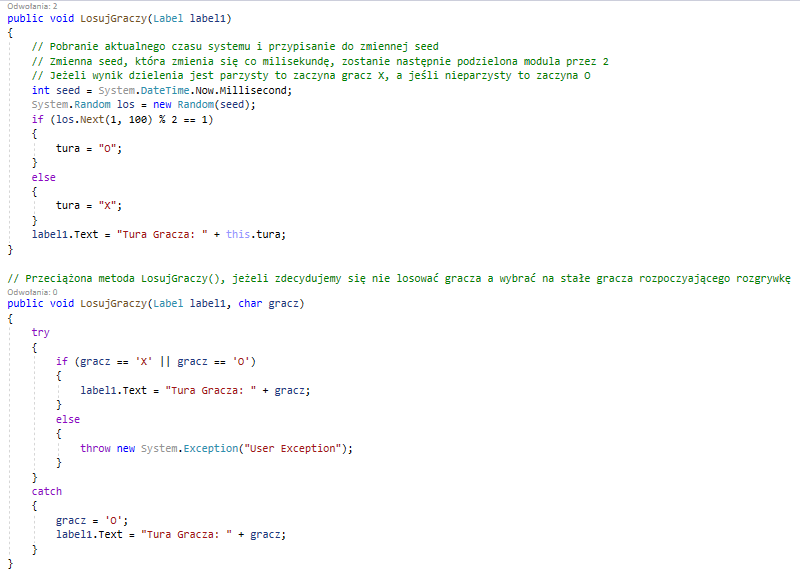
* **własne klasy z odpowiednio ustawionym poziomem dostępu do jej składowych,**
* **mechanizm dziedziczenia klas,**
* **zastosowanie konstruktora w klasie macierzystej i potomnej,**

Aplikacja składa się 2 kluczowych klas: Gracze i MistrzGry. Obie zostały opisane w komentarzach w kodzie źródłowym



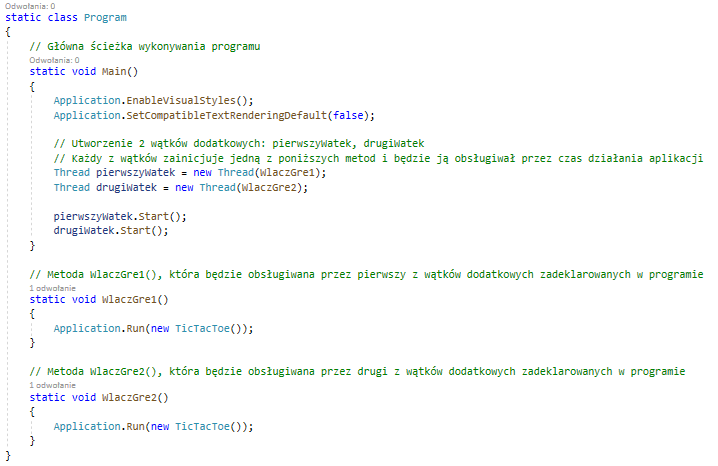
Klasa potomna:

* **dowolny mechanizm polimorficzny,**
* **przechwycenie i obsługa co najmniej jednego wyjątku w ramach aplikacji;**



**Wykorzystanie poznanych mechanizmów sieciowych i/lub wątków (wystarczy wybrać jedno z poniższych):**

* **wykorzystanie osobnego wątku lub wątków w aplikacji;**

****

**Stosowanie konwencji nazewniczych w programowaniu i formatowania kodu typowego dla wybranej technologii:**

* **konwencja nazewnictwa w ramach klas (klasa, zmienne, metody, właściwości) i interfejsów,**
* **(w przypadku projektów z obsługą zdarzeń) konwencja nazewnictwa w ramach zdarzeń,**
* **formatowanie kodu – wcięcia i odstępy w stosownych miejscach;**

W trakcie tworzenia aplikacji zachowaliśmy konwencje nazewnicze i wcięcia w kodzie co widać w kodzie źródłowym naszej aplikacji.