# **Specyfikacja implementacyjna:** `Kwantowe Kółko i Krzyżyk`

**Michał Żdanuk**


Wydział Elektryczny
Politechnika Warszawska
Marzec 2022

# 1 Cel dokumentu

Dokument powstał, aby ułatwić zaprogramowanie gry w **Kwantowe Kółko i Krzyżyk**. Przybliży on ideę tworzenia programu wraz ze szczegółami implementacyjnymi.

Dokument zawiera diagram klas wraz z ich charakterystykami, które zdefiniowałem w celu realizacji wszystkich wymagań programu. Oprócz tego w specyfikacji znajduje się krótka charakterystyka środowiska pracy, opis tego, w jaki sposób będą prowadzone prace projektowe oraz wyjaśnienie, w jaki sposób będę testować działanie programu.

*Dokument ten związany jest ze **specyfikacją funkcjonalną**.*

# 2 Wstęp teoretyczny

Celem projektu jest zaprogramowanie gry, która pozwoli dwóm graczom (przy jednym urządzeniu) stoczyć pojedynek w wszystkim dobrze znanej grze jakim jest Kółko i Krzyżyk, jednakże w wersji kwantowej. Początkowo gra zostanie wykonana w wersji terminalowej(wyświetlanie planszy oraz zaznaczanie pól będzie wykonywane przy pomocy terminalu). Po skończeniu wersji "terminalowej" gra będzie rozwinięta o interfejs graficzny (GUI).

*Kompleksowy opis zasad gry "Kwantowe Kółko i Krzyżyk" z wszelkimi szczegółami znajduje się w **specyfikacji-funkcjonalnej-proj-ind**.*

# 3 Środowisko pracy

W projekcie pracował będę na własnym stanowisku roboczym w swoim domu. Poniżej prezentuję krótką specyfikację systemu i urządzenia na którym będę pracował do realizacji gry:

- Windows 10 Home, Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz, pamięć RAM 6GB, środowisko Visual Studio Code

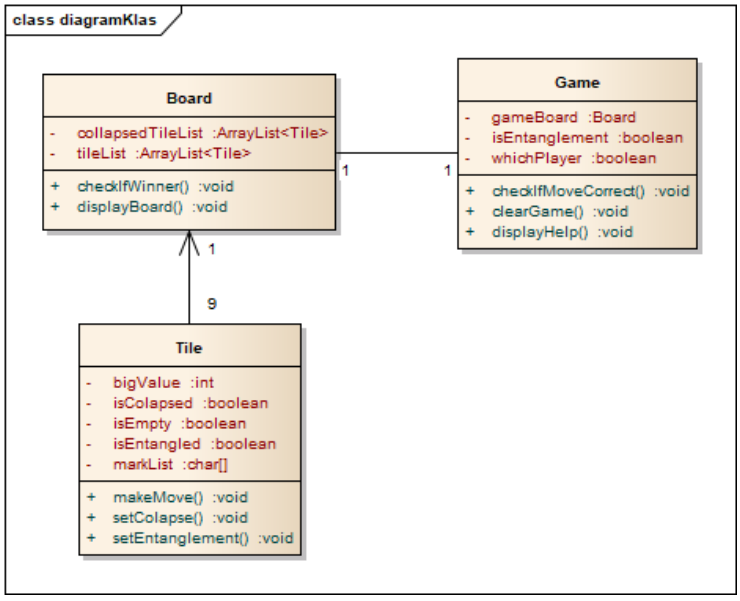
Wszelka dokumentacja powstająca przy tworzeniu projektu pisana jest z wykorzystaniem języka publikacyjnego `Latex` za pomocą narzędzia `Overleaf`. Diagram zaprezentowany w tym dokumencie został utworzony przy pomocy narzędzia `Enterprise Architect` na licencji studenckiej.

# 4 Opis zdefiniowanych klas

W celu stworzenia gry powstaną niżej wymienione klasy:

- Tile - najmniejszy fragment systemu. Zawiera listę, w którą można wpisywać zaznaczane znaki. Posiada kilka flag: isEmpty(informująca o tym czy nie wpisano jeszcze żadnego "znaczku" w komórkę), isEntangled(informująca o tym czy komórka jest w stanie `"splątania"`), isColapsed(gdy flaga jest zapalona oznacza to, że w komórce znajduje się "duży znaczek" i nie można już edytować zawartości tej komórki).
- Board - klasa zbudowana z komórek. Zawiera listę komórek, które uległy `"zawaleniu"`. Ma metody do wyświetlania planszy w okienku terminalu, sprawdzenia czy któryś z graczy wygrał po wykonanym ruchu.
- Game - klasa posiadająca planszę(Board), służąca do obsługi gry. Ma informację o tym czy w aktualnej chwili gry wystąpiło `splątanie` lub `zawalenie się` komórek. Posiada także metody do sprawdzenia, czy wykonywany ruch jest poprawny, w przypadku nieprawidłowego wyświetla komunikat o błędzie wraz z wskazówką w jaki sposób wykonać poprawne posunięcie.

# 5 Diagram klas




Rysunek 1: Diagram klas projektu

# 6 Forma prowadzenia projektu

Pracę w projekcie będę wykonywać, działając na zdalnym repozytorium GIT'a. Poniżej zamieszczam link do repozytorium:

- https://github.com/MichalZdanuk/Quantum-TicTacToe

Planuję, by prace były prowadzone systematycznie (tzn. co najmniej raz w ciągu dwóch tygodni w repozytorium będzie się pojawiać nowy fragment kodu, działającej części projektu).

# 7 Metody testowania programu

Testy jednostkowe metod zostaną przeprowadzone za pomocą biblioteki `JUnit`. Wykorzystane zostaną możliwości IDE, aby łatwo testować kod za pomocą specjalnych pakietów testowych. Będę chciał, aby testy były robione na bieżąco - tj. powstawały wraz z tworzonymi metodami. Planuję możliwie jak największe pokrycie kodu testami.

Poza automatycznymi testami jednostkowymi przeprowadzę testy statyczne (tj. przeglądanie kodu w poszukiwaniu błędów) oraz testy dynamiczne - obserwując zachowanie programu "manualnie".