# ConnectFour (Cztery w rzędzie)

*Cztery w rzędzie* (*Connect Four*) planszowa gra logiczna dla dwóch osób, w której wykorzystuje się planszę o wymiarach 7 x 6 pól. Pierwszy gracz wrzuca swój żeton do wybranej przez niego kolumny. Żeton zajmuje najniższą pozycję. Gracze wrzucają swoje żetony na przemian, aż jeden z nich ułoży cztery żetony w poziomie, pionie lub ukosie. Wygrywa ten gracz, który zrobi to jako pierwszy. Jeżeli natomiast plansza się zapełni, a nie utworzy się żadna czwórka, jest remis. Wikipedia Program implementuje zarówno klasyczną wersję gry jak i odmianę *PopOut* z możliwością usuwania żetonów ze spodu planszy.

## Technologie

- Python 3.8.2
- PySide2 5.15.2
- Qt 5.15.2

## Przegląd aplikacji

Okno wyświetla siatkę 7 kolumn x 6 wierszy, przyciski nad (do zrzucania żetonów) i pod (do wyciągania żetonów) każdą kolumną, informacje o stanie rozgrywki (np. "*Player 1 turn!*"), przycisk do rozpoczynania i resetowania gry oraz rozwijalną listę wybory reguł gry. Początkowo pola siatki są puste. Gracze na zmianę wrzucają monety do wybranych przez siebie kolumn. Pola, w których jest żeton gracza 1 są czerwone, pola z żetonami gracza 2 są żółte. Gracze wybierają kolumnę, do której wrzucą żeton klikając przycisk nad nią lub (w przypadku *PopOut*) pod nią, by usunąć swój żeton ze spodu planszy. Wygrywa gracz, który pierwszy ustawi cztery monety w linii (poziomo, pionowo lub po skosie). Gdy gra się skończy, wyświetlane jest okienko z napisem "*Player 1 won!*", "*Player 2 won!*" lub "*Game drawn!*". Możliwe jest zresetowanie planszy bez zamykania głównego okna.

## Przegląd kodu

Odnośniki do kluczowych fragmentów kodu znajdują się w zakładce **Istotne fragmenty kodu**.

Za logikę gry odpowiadają klasy modelujące rozgrywkę: klasa `ConnectFourBase` jest klasą bazową dla implementacji kompletnych reguł w klasach `ConnectFourClassic`, która implementuje klasyczną wersję gry *ConnectFour* i `ConnectFourPopOut`, która implementuje zmodyfikowaną wersję gry - *PopOut*:

```
class ConnectFourBase
class ConnectFourClassic(ConnectFourBase)
class ConnectFourPopOut(ConnectFourBase)
```

Rozgrywka toczy się w oknie głównym `MainWindow`, które wyświetla planszę, przyciski do gry, startowania i resetowania rozgrywki, listę rozwijalną z dostępnymi trybami rozgrywki oraz indykator statusu gry. Do wyświetlania ważnych komunikatów w trakcie rozgrywki, takich jak wygrana jednego z graczy lub remis, używane jest modalne okienko dialogowe `GameStateDialog`:

```
class  MainWindow(QMainWindow)
class  GameStateDialog(QDialog)
```

Do obsługi błędnego ruchu użytkownika w trakcie gry używana jest klasa wyjątku `WrongMoveException`. Wyjątek ten rzucany jest za pośrednictwem zaprezentowanych wyżej klas modelujących rozgrywkę (np. `ConnectFourClassic`) oraz obsługiwany z poziomu kodu implementującego graficzny interfejs użytkownika (`MainWindow`):

```
class  WrongMoveException(Exception)
```

## Testy

Projekt zawiera następujące testy przeprowadzone za pomocą modułu `unittest`:

1. `test_should_board_contain_two_coins_when_two_drops` Wykonanie po dwa ruchy przez każdego z graczy - monety spadają na dół pola gry lub zatrzymują się na już wrzuconym żetonie.
2. `test_should_return_true_when_vertical_win_line` Ułożenie pionowej linii monet przez jednego gracza - oczekiwana informacja o jego wygranej.
3. `test_should_return_true_when_horizontal_win_line` Ułożenie poziomej linii monet przez drugiego gracza - oczekiwana informacja o jego wygranej.
4. `test_should_return_true_when_diagonal_win_line` Ułożenie skośnej linii przez dowolnego gracza - oczekiwana informacja o jego wygranej.
5. `test_should_return_true_when_board_tied` Zapełnienie pola gry tak, że żaden gracz nie ułożył linii - oczekiwana informacja o remisie.
6. `test_should_return_true_when_longer_than_four_win_line` Ułożenie linii dłuższej niż 4 przez jednego z graczy - oczekiwana informacja o jego wygranej.
   [c][c][c][ ][c][c][c]
   [ż][ż][ż][ ][ż][ż][ż] <- w następnym ruchu gracz żółty wrzuci monetę w środkową kolumnę.
7. `test_should_throw_wrong_move_exception_when_wrong_drop` Próba wrzucenia monety do zapełnionej kolumny - oczekiwana informacja o błędzie.

Potwierdzenie zgodności kodu z testami:

```
.......
----------------------------------------------------------------------
Ran 7 tests in 0.001s
```

```
OK
```

## Podsumowanie

Projekt udało się w pełni zrealizować. Wytyczne dla projektu zostały wypełnione i rozwiązanie jest w pełni kompletne. Testy zawarte w pliku `connectfour_test.py` potwierdzają działanie logiki programu.

## Istotne fragmenty kodu

1. Wyrażenia lambda:
   - Przykład 1

```
for drop_button, pop_button, column in zip(self.drop_buttons, self.pop_buttons, range(7)):
    drop_button.clicked.connect(lambda *args, column=column: self.drop_move(column))
    pop_button.clicked.connect(lambda *args, column=column: self.pop_move(column))
```

   - Przykład 2

```
self.game_mode_combo_box.currentIndexChanged.connect(
    lambda *args: self.set_game_mode(self.game_mode_combo_box.currentText()))
```

   - Przykład 3

```
self.ok_button.clicked.connect(lambda: self.accept())
```

2. Wyrażenia listowe:
   - Przykład 1

```
positions = list(chain.from_iterable([[(i, j) for j in range(7)] for i in range(6)
```

   - Przykład 2

```
return [[0 for i in range(self.column_count)] for i in range(self.row_count)]
```

   - Przykład 3

```
expected_board = [[0 for i in range(7)] for i in range(6)]
```

3. Klasy:

- Przykład 1

```
class  ConnectFourBase
```

- Przykład 2

```
class  ConnectFourClassic(ConnectFourBase)
```

- Przykład 3

```
class  ConnectFourPopOut(ConnectFourBase)
```

- Przykład 4

```
class  MainWindow(QMainWindow)
```

- Przykład 5

```
class  GameStateDialog(QDialog)
```

4. Wyjątki:

- Przykład 1

```
class  WrongMoveException(Exception)
```

- Przykład 2

```
try:
    move_func(self, *args, **kwargs)
    self.render_board()
except WrongMoveException:
    self.info_label.setText("Can't make a move here!")
```

5. Moduły:

```
ConnectFour/
│
├── drawable/
│   ├── empty_field.png
│   ├── red_field.png
│   ├── yellow_field.png
├── gamemodel/
│   ├── __init__.py
│   ├── connectfour.py
│   ├── wrongmoveexception.py
├── ConnectFour.pyproject
├── README.md
```

```
├── __init__.py
├── connectfour_test.py
├── gamestatedialog.py
└── mainwindow.py
```

6. Dekoratory:

- Przykład 1

```
def move(move_func):
'''A wrapper function for any type of move during the game'''

    def wrap(self, *args, **kwargs):
        try:
            move_func(self, *args, **kwargs)
            self.render_board()
        except WrongMoveException:
            self.info_label.setText("Can't make a move here!")
        return wrap

@move
def drop_move(self, column):
    # drop_move implementation (throws WrongMoveException)

@move
def pop_move(self, column):
    # pop_move implementation (throws WrongMoveException)
```