Białystok 25.01.2022r.

Politechnika Białostocka

Projekt: Kółko i krzyżyk

Wydział Informatyki

Grupa zajęciowa: PS 8

Wykonawcy:

Konrad Augustynowicz

Paweł Augustyniak

Mateusz Nowosadko

Mateusz Wiszowaty

Prowadzący:

dr inż. Jerzy Krawczuk

1. Opis i cel projektu

Tematem naszego projektu jest program umożliwiający grę w kółko i krzyżyk przez dwóch graczy przez sieć. Aplikacja umożliwia rozgrywkę w momencie, w którym obydwu graczy połączy się z serwerem.

2. Opis funkcjonalności programu

Projekt dzieli się na dwa programy:

1 Serwer

-obsługuje jedną grę 2 graczy w danym czasie

-synchronizuje stan rozgrywki użytkowników oraz śledzi czyja jest aktualnie tura.

-przyjmuje informacje o ruchach wykonanych przez graczy oraz odpowiednio ją obsługuje.

-po uruchomieniu oczekuje na połączenie dwóch klientów, po skończonej rozgrywce ponownie przyjąć dwóch graczy i rozpocząć grę.

2 Klient

-posiada okno z grą, za pomocą którego prowadzi rozgrywkę

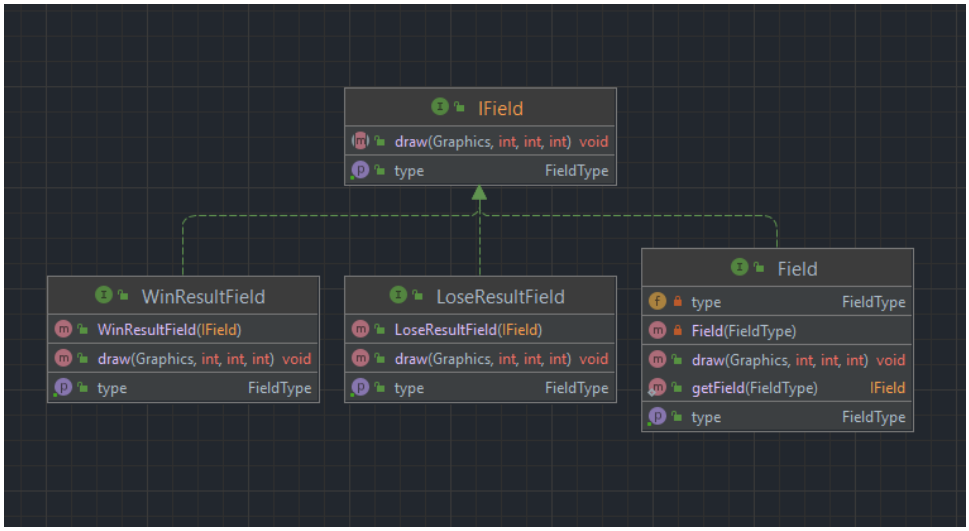
-umożliwia graficzną interakcje z planszą, wybieranie pola poprzez klikanie myszką.

-umożliwia odtworzenie przebiegu rozgrywki

-pozwala pauzować rozgrywkę, wznowić grę może gracz, którą ją zatrzymał.

3. Indywidualny opis każdego wzorca

-Dekorator

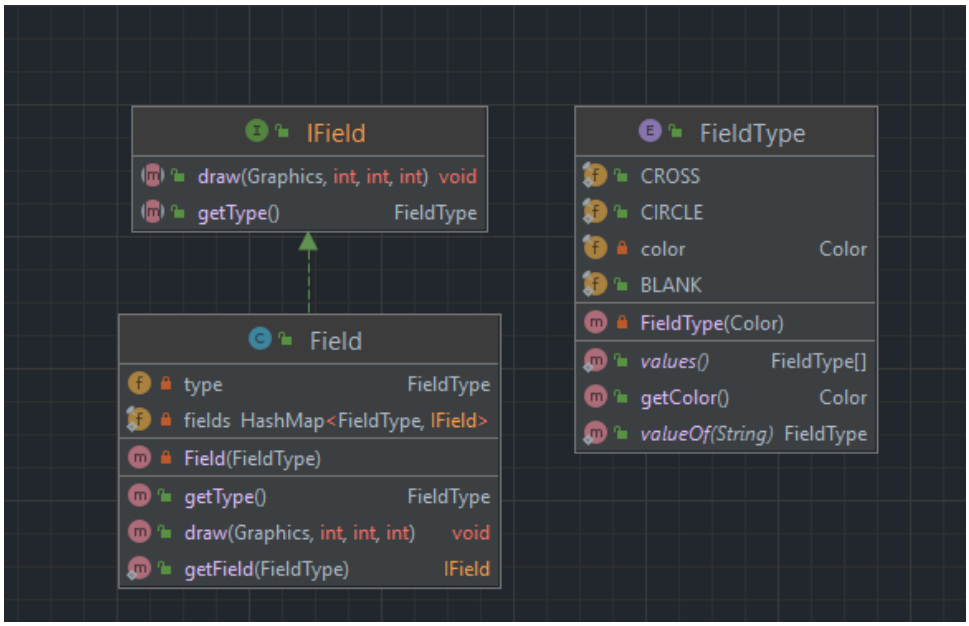


Dekorator służy do zmiany sposobu rysowania pola w zależności od sytuacji tzn. Czy dany gracz wygrał czy przegrał to zaznaczane są pola, które skończyły grę. Pozwala na przyjemniejsze zmienianie zachowanie metod obiektów bez wielopiętrowych konstrukcji if. Wektor zmian – dynamiczna zmiana zachowania obiektu.

Zastosowanie w kodzie, po otrzymaniu wiadomości o wygranej:

```
fields.setAt(y, x, new WinResultField(fields.getAt(y, x)));
```

-Pyłek



Wzorzec pyłek wykorzystano do zmniejszenia kosztu pamięciowego przechowywania wielu instancji identycznych obiektów. Statyczne pole field i metoda getField zarządza instancjami obiektów tworząc lub zwracając już istniejącą instancje obiekty o określonym “FieldType”. Wektor zmian – koszt przechowywania wielu takich samych małych obiektów. Zastosowanie przy inicjalizacji planszy, we wszystkie pola wstawiana jest ta sama instancja obiektu Field:

```
for (int j = 0; j < matrix[i].length; ++ j) {
    matrix[i][j] = Field.getField(FieldType.BLANK);
}
```

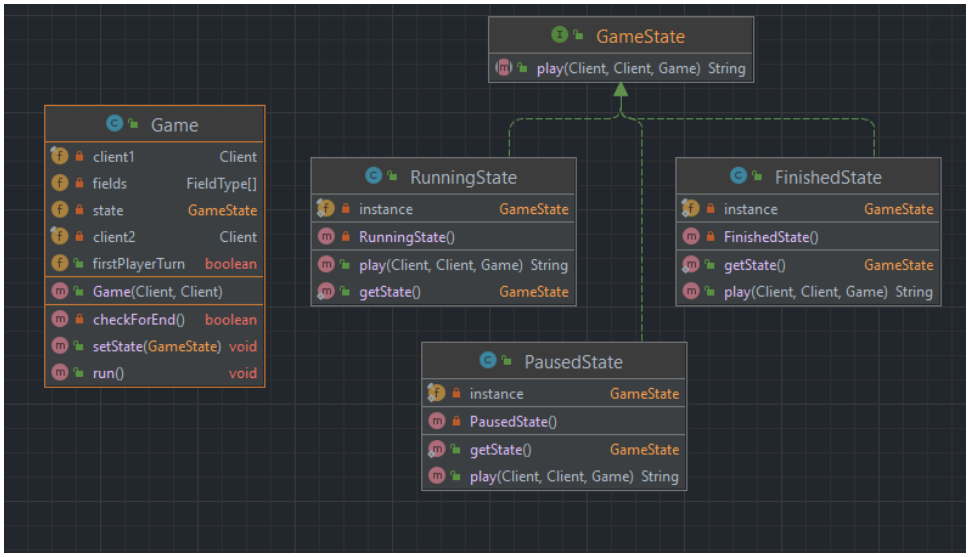
Stan zewnętrzny (pozycja i szerokość) są przekazywane do metod z zewnątrz. Stanem wewnętrznym jest wartość enuma FieldType określająca czy pole jest puste, z kółkiem lub z krzyżykiem.

Wywołanie metody draw na obiekcie Field:

```
matrix[y][x].draw(g, x, y, tilesize);
```

-Stan

Stan używany jest do kontroli przepływu gry. Określa jakie wiadomości są wysyłane do graczy oraz jakie i w jaki sposób są przyjmowane. Zmienia on reakcje pętli gry na otrzymywane wiadomości od graczy oraz zarządza zmianą stanu.



Zmiana stanu gry na zatrzymaną po otrzymaniu wiadomości PAUSE:

```
msg = state.play(client1, client2, game: this);

String[] split = msg.split( regex: ";");
if (msg.equals("PAUSE") && !(state instanceof PausedState))
    state = PausedState.getState();
```

Zmiana stanu gry na zwykły po otrzymaniu kolejnej wiadomości PAUSE w metodzie wykonywanej przez stan PausedState:

```
if(msg.equals("PAUSE")){
    game.setState(RunningState.getState());
}
```

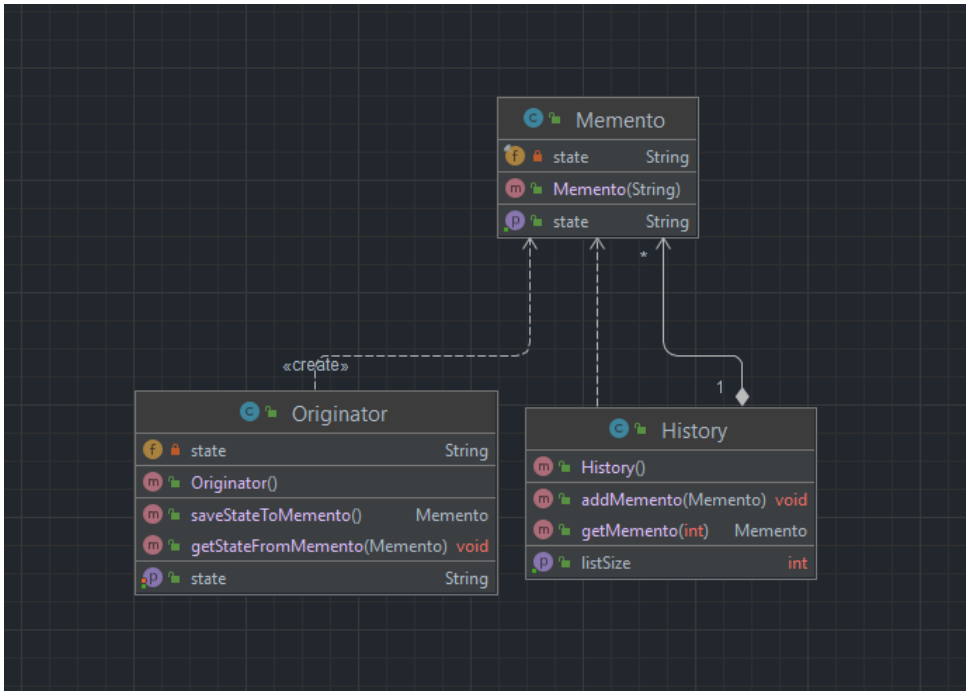
-Singleton

Poszczególne klasy implementujące interfejs GameState są singletonami. Nie potrzebne są wielokrotne instancje tych klas. Klasa posiada prywatny konstruktor, pole statyczne przechowującą jedyną instancje oraz metode statyczną, która ją udostępnia. Jedyne wywołanie tej metody znajdują się w klasie Game serwera.

W konstruktorze klasy Game stan ustawiany jest na RunningState a po otrzymaniu sygnału pauzy zmienia się na PausedState do momentu aż gracz nie wznowi gry.

```
    state = RunningState.getState();
}
```

- Memento



Wzorzec memento użyto w celu przechowywania historii ruchów użytkowników na polu gry. Po zakończonej rozgrywce gracze mają możliwość prześledzić wykonane przez siebie i przeciwnika ruchy podczas gry. Originator zapisuje i odtwarza swój stan w postaci obiektu Memento. Obiekt Caretaker (nazwa: History) przechowuje obiekty Memento, ale nie ma dostępu do ich danych.

4. Instrukcja instalacji

Aby grać potrzebny jest działający serwer na jednym komputerze. Następnie podając jego zewnętrzny adres IP oraz port gracze mogą się do niego łączyć. Gra zaczyna się, gdy podłączyło się dwóch użytkowników. Uruchomienie aplikacji użytkownika automatycznie próbuje się połączyć z serwerem pod podanym adresem.

5. Instrukcja użytkownika

Gra rozpoczyna się w momencie, w którym dwóch graczy połączy się z serwerem oraz wciśnie przycisk start. Gracz, który połączy się jako pierwszy będzie również tym, którego tura będzie pierwsza. Pola planszy podświetlają się tylko i wyłącznie jeśli aktualnie jest twoja tura oraz nie ma na nich żadnego znaku. W momencie, w którym gra zostanie ukończona poprzez wygraną lub remis zostaje wyświetlona odpowiednia informacja(np. o wygranej).

6. Podział pracy w zespole

Paweł Augustyniak - wzorce pyłek, dekoratory, stan, singleton, logika gry

Mateusz Wiszowaty - wzorzec memento, mechanizm odtwarzania przebiegu rozgrywki, logika gry

Konrad Augustynowicz – interfejs, komunikacja przez gniazda, logika gry

Mateusz Nowosadko – interfejs, logika gry

7. Rozwiązanie specyficzne dla użytej technologii

Każda sytuacja, która wymaga reakcji serwera na wiadomość od gracza wykorzystuje oddzielny wątek z pętlą działającą, dopóki przez wejście gniazda nie otrzymamy odpowiedniej informacji. Również sam wątek gry działa w pętli kończącej się dopiero po wyjściu z programu wczytując wszystkie dane jakie serwer przesyła klientom i odpowiednio reagując.