Sprawozdanie z projektu aplikacji współbieżnej

Protokół komunikacyjny oraz implementacja gry Achtung, die Kurve!

Gracjan Grzech
193 579

Hubert Wajda
193 511

Artem Dychenko
192 441

1. Wprowadzenie

Sprawozdanie dotyczące projektu aplikacji bazującej na popularnej grze wieloosobowej Achtung, die Kurve. Zawiera model komunikacji, podstawowy schemat działania, wraz z definicją protokołu komunikacji między graczami, a serwerem, oraz z zaprezentowanym diagramem sekwencji.

2. Komunikacja

Wybrany przez nas protokół warstwy transportowej internetu to TCP. Zapewnia on pewnosc dostarczenia wiadomości w takiej kolejności, w jakiej została ona wysłana, kosztem prędkości oraz wielkości paczki. Z uwagi na fakt, ze wysyłane przez nasz serwer wiadomości są małej wielkości, utrata prędkości powinna być mała. Natomiast niezawodność protokołu TCP znacznie ułatwia implementację gry.

Protokół warstwy aplikacji wykorzystany w implementowanej grze to WebSocket, do którego API Java udostępnia w klasie `Socket`. Wiadomości odbierane i wysyłane będą za pomocą `ObjectInputStream` oraz `ObjectOutputStream`. Zastosowanie tych klas pozwala na automatyczną serializację i deserializację obiektów tworzących zawartość wiadomości.

Poniższe tabele prezentują dokladną strukturę każdego możliwego komunikatu:

2.1. Wiadomosci serwera

Aktualizuj Stan Gry (Serwer → Klient)		
Nazwa pola	**Typ danych**	**Opis**
moves	List<Pair<Gracz.Color,Pole>>	Ruchy innych graczy
isGameOver	bool	Czy po wykonanym ruchu gracz zyje

Rozpocznij gre (Serwer → Klient)		
Nazwa pola	**Typ danych**	**Opis**
size	Pair<int, int>	Rozmiar planszy

Zakonczenie gry (Serwer → Klient)		
Nazwa pola	**Typ danych**	**Opis**
points	int	Punkty zdobyte za obecna runde

2.2. Wiadomosci klienta

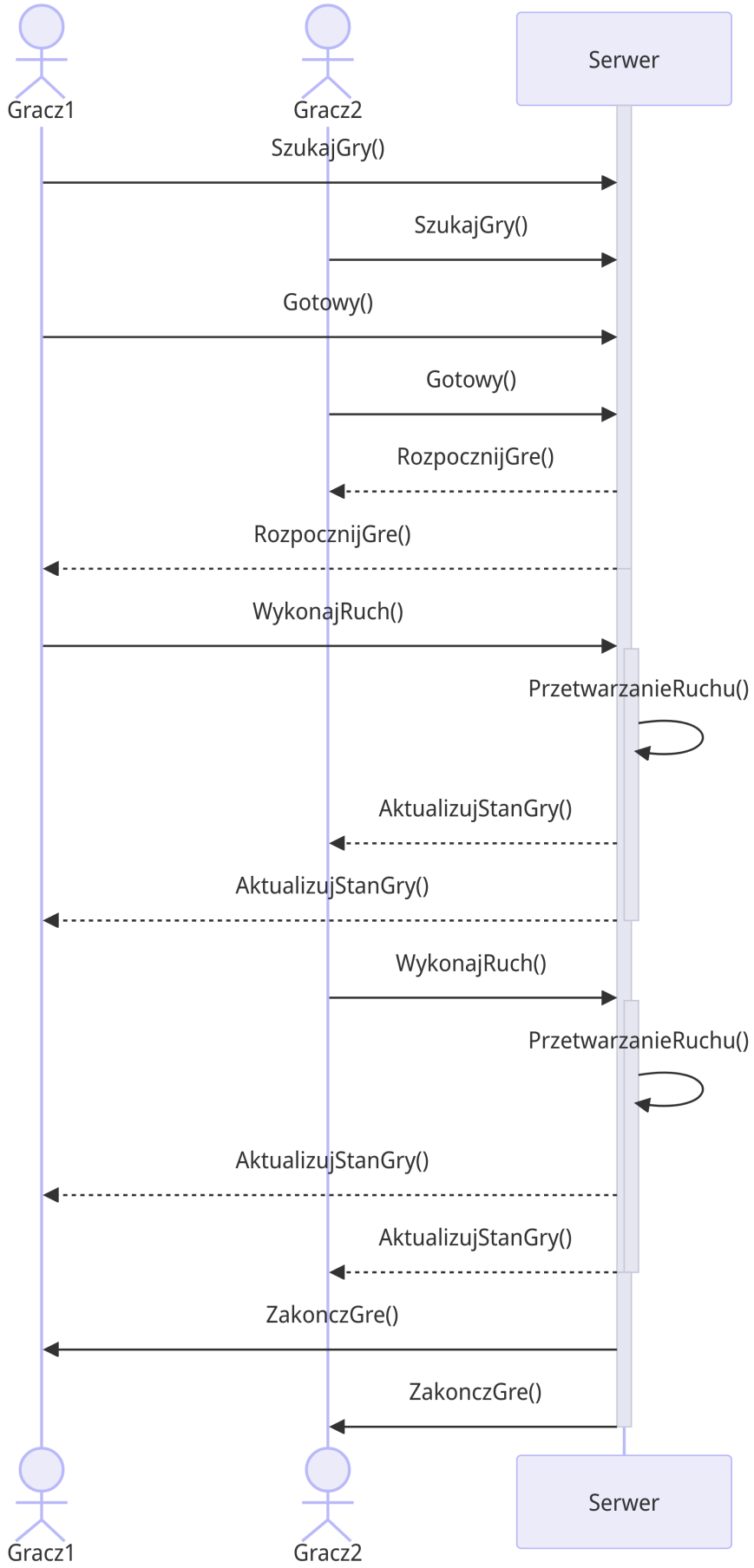
Szukanie Gry (Klient → Serwer)		
Nazwa pola	**Typ danych**	**Opis**
nick	string	Nazwa gracza

Wykonaj Ruch (Klient → Serwer)		
Nazwa pola	**Typ danych**	**Opis**
ready	string	Gracz zglasza swoja gotowsc do gry

Zgloś gotowość (Klient → Serwer)		
Nazwa pola	**Typ danych**	**Opis**
move	string	Wykonany ruch - “LEWO”, “PRAWO”, “GÓRA”, DÓŁ

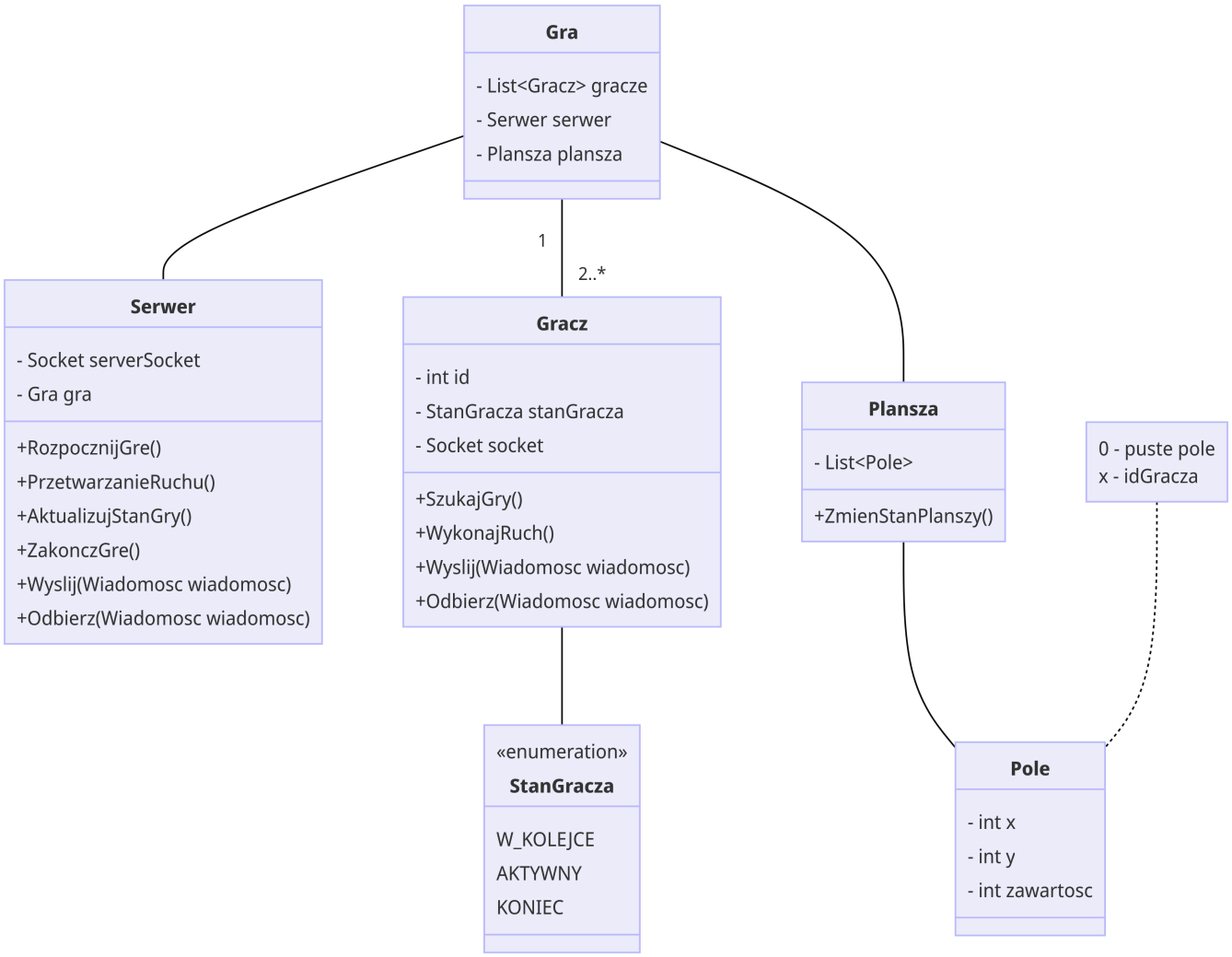
2.3. Diagram Sekwencji

Na następnej stronie znajduje się diagram sekwencji prezentujący naturalny przebieg gry. Rozgrywka zaczyna się od dołączenia graczy do poczekalni. W tym celu klient wysyła komunikat `SzukajGry()`. Po dołączeniu pewnej ilości graczy, graczy po kolei wysyłają do serwera komunikat `Gotowy()`, po wysłaniu odpowiedniej ilości komunikatów, co jest graczy w poczekalni(od 2 do 6), serwer informuje klientów o rozpoczęciu gry, losując startowe położenia graczy i wysyłając komunikat `RozpocznijGrę()` ze startowymi koordynatami każdego gracza. Po rozpoczęciu gry serwer w stałych odstępach czasu wysyła klientom zmiany aktualnego stanu planszy. Jako że gracz porusza się bez przerwy, klient nie ma potrzeby przesyłania informacji o swoim przemieszczeniu, wysyła on komunikat tylko w momencie zmiany kierunku, w którym się porusza. Kiedy wszyscy gracze na planszy zostaną wyeliminowani, serwer informuje o tym klientów, wysyłając komunikat `ZakończGrę()`. Gracze grają trzy rundy, w których punkt przyznawany tylko za wygranie rundy. Zwycięża tem gracz który po 3 rundach będzie mial ich więcej.



rys. 1 Diagram Sekwencji

2.4. Diagram Klas



rys. 2 Diagram Klas

Powyższy diagram klas prezentuje strukture modeli potrzebnych do implementacji logiki gry:

- Gra - reprezentuje samą grę. Ma listę graczy, serwer oraz planszę. Jest odpowiedzialna za koordynację gry oraz zarządzanie stanem gry
- Gracz - reprezentuje gracza. Ma swoje identyfikator id, stan gracza oraz gniazdo socket do komunikacji z serwerem. Odpowiada za wyszukiwanie gry, wykonywanie ruchów oraz obsługę komunikacji z serwerem
- Plansza - odpowiada za zarządzanie polami planszy oraz zmianę stanu planszy
- Pole - pojedyncze pole mapy
- Stan gracza - stany w jakich może znajdować się gracz
- Serwer - zarządza połączeniami sieciowymi i obsługuje komunikcję między graczami. Odpowiada za rozpoczęcie gry, przetwarzanie ruchów graczy, aktualizację stanu gry oraz zakończenie gry

3. Elementy krytyczne

Fundamentalnym aspektem naszej gry jest obiekt ‘Gra’, który przechowuje liste graczy oraz aktualny stan planszy. Gracze poruszają się bez przerwy, zatem do serwera wysyłają tylko i wyłącznie zmiany kierunku. Plansza zmienia swój stan co określony czas (0,5s). Kluczowe jest aktualizowanie po kolei dodatkowcyh położeń graczy na planszy oraz sprawdzanie kolizji po każdym “cyklu” wykonania ruchu.

3.1. Gracz traci połączenie

Każdy gracz regularnie wysyła wiadomość keepAlive w ustalonych odstępach czasu(0.5s). Jest to sposób na sprawdzenie, czy dany gracz nadal aktywnie uczestniczy w grze. W przypadku, gdy gracz zostanie rozłączony, uznaje się go za wyeliminowanego, a jego wynik w grze zostaje wyzerowany. Jeśli wszyscy gracze zostaną rozłączeni jednocześnie, prowadzi to do remisu.

3.2. Gracze uderzają siebie nawzajem jednocześnie

W przypadku takiego zachowania dwóch graczy, gra kończy się dla nich remisem. Jeśli ci dwaj gracze byli ostatnimi uczestnikami gry, nie ma zwycięzcy, więc wynik gry jest ustawiony na remis.