Dokumentacja techniczna

Rozproszona platforma do tworzenia gier planszowych, przeznaczona na system Android.

*Niniejsze opracowanie powstało w trakcie i jako rezultat za­jęć dy­dak­tycz­nych z przedmiotu wymienionego na stronie tytułowej, pro­wa­dzo­nych w Akademii Górniczo-Hutniczej w Krakowie (AGH) przez oso­bę (oso­by) wy­mie­nioną (wymienione) po słowach "Pro­wa­dzą­cy zajęcia" i nie może być wy­korzystywane w jakikolwiek sposób i do jakichkolwiek ce­lów, w ca­ło­ści lub części, w szczególności pub­li­ko­wa­ne w ja­ki­kol­wiek spo­sób i w jakiejkolwiek formie, bez uzy­ska­nia uprzed­niej, pi­sem­nej zgody tej oso­by (tych osób) lub odpowiednich władz AGH.****Copyright © 2013 Akademia Górniczo-Hutnicza (AGH) w Krakowie***

**Spis treści**

1. Zakres dokumentu 3

2. Projekt klas 4

2.1.1. Pakiet game 4

2.1.2. Pakiet gameResources 8

2.1.3. Pakiet gameInitializers 12

2.1.4. Pakiet gameUtilites 15

2.1.5. Pakiet network 16

2.1.6. Pakiet platformExceptions 19

2.1.7. Pakiet tests 20

3. Bibliografia 21

Indeks ważniejszych terminów 22

Spis ilustracji i tabel 22

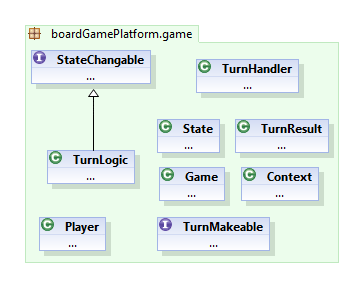
# Zakres dokumentu

Dokument ten ma na celu przedstawienie interfejsu programistycznego platformy. Jego głównym celem jest umożliwienie twórcy gry zapoznania się ze sposobem realizacji podstawowych założeń platformy.

Dokument jest uzupełnieniem dokumentacji zawartej w JavaDoc platformy.

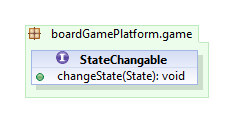
# Projekt klas

### Pakiet game



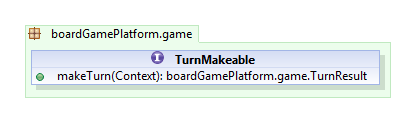
Interfejsy:

1. StateChangable



Interfejs reprezentujący obiekt którego stan może być zmieniany.

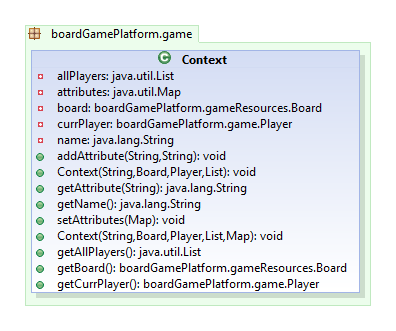
1. TurnMakeable



Interfejs umożliwiający wykonanie tury na podstawie zadanego kontekstu. Klasa implementująca musi specyfikować w jaki sposób ma zostać wykonana tura, tj. jakie komunikaty maja pojawić sie u gracza, oraz w jaki sposób (jeżeli to konieczne) ma zostać pobrane wejście.

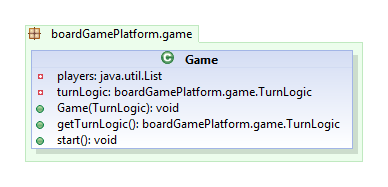
Klasy:

1. Context



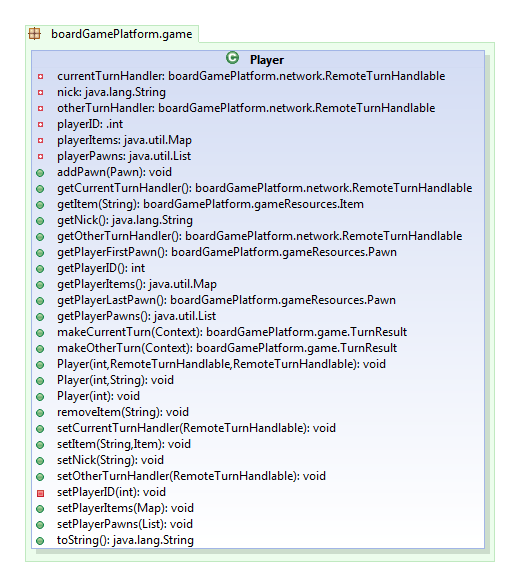
Klasa reprezentująca obecny stanu gry przekazywany do wszystkich graczy. Obiekty są rozróżnialne na podstawie nazwy. Zawiera obecna plansze, informacje o graczu aktualnie wykonującym ruch oraz o wszystkich graczach biorących udział w rozgrywce. Pozwala także na przechowywanie dodatkowych atrybutów reprezentowanych za pomocą łańcucha znaków oraz rozróżnialnych przez nazwę. Zawiera odpowiednie mechanizmy dostępu do wszystkich informacji.

1. Game



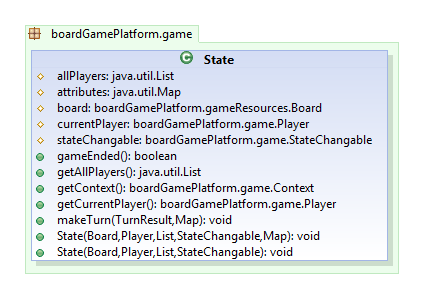
Głowna klasa zarządzająca całym przebiegiem rozgrywki. Do poprawnego działania wymaga podania logiki gry, która ma obsługiwać oraz pełnej listy graczy biorących udział w rozgrywce.

1. Player



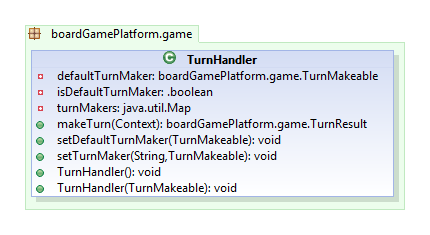
Klasa reprezentująca obiekt gracza biorącego udział w rozgrywce. Zawiera wszystkie elementy niezbędne do komunikowania sie z graczem, informowania go o obecnym stanie rozgrywki oraz pobierania od niego wejścia, w razie potrzeby. Poza tym przechowuje informacje o jego nicku, przedmiotach dodatkowych i pionkach które zostały do niego przypisane.

1. State



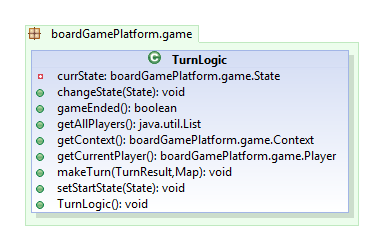
Klasa abstrakcyjna, która jest implementowana przez twórcę gry. Zawiera ona informacje o akcjach które musza zostać wykonane w momencie gdy gra znajdzie sie w danym stanie, oraz o tym w jakim stanie znajdzie sie po wykonaniu zdefiniowanych akcji. Na potrzeby dokonania koniecznych zmian obiekty klasy przechowują informacje o obecnym stanie planszy, graczu którego obecnie jest tura, oraz wszystkich graczach biorących udział w rozgrywce. Istnieje także możliwość przekazania dodatkowych informacji koniecznych do dokonania zmian w dodatkowym parametrze attributes. Klasa ta jest wykorzystywana przez obiekt StateChangable do zarządzania przebiegiem gry. Każda prawidłowa implementacja gry, z wykorzystaniem platformy musi zawierać co najmniej jedna implementacje klasy State - stan początkowy, który określa co ma sie stać w momencie rozpoczęcia rozgrywki.

1. TurnHandler



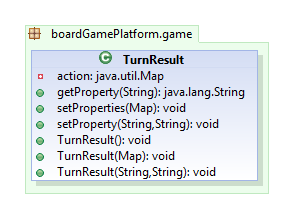
Klasa zajmująca sie przechowywaniem odwzorowań nazw kontekstów na obiekty zajmujące sie obsługą tur, których one dotyczą, oraz oddelegowywaniem wykonania tury do odpowiednich obiektów implementujących TurnMakeable. Pozwala na ustawienie domyślnego obiektu zajmującego sie obsługą tur.

1. TurnLogic



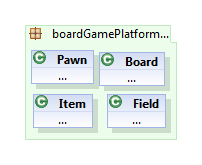
Klasa implementująca StateChangable. Pozwala na ustawienie stanu początkowego rozgrywki oraz przechowuje jej stan obecny. Daje także możliwość wykonania tury dla obecnego stanu oraz sprawdzenia czy gra dobiegła końca.

1. TurnResult



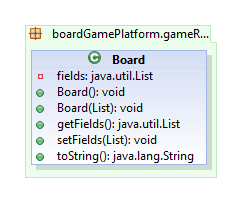
Klasa przechowująca rezultat tury za pomocą mapy nazw akcji oraz ich wartości przechowywanych jako łańcuchy znakowe.

### Pakiet gameResources



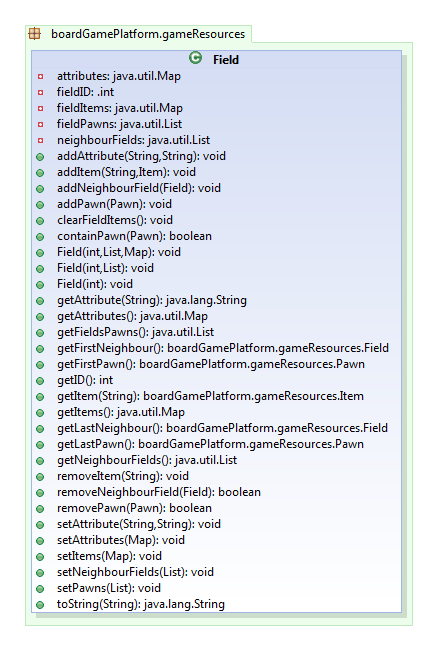
Klasy:

1. Board



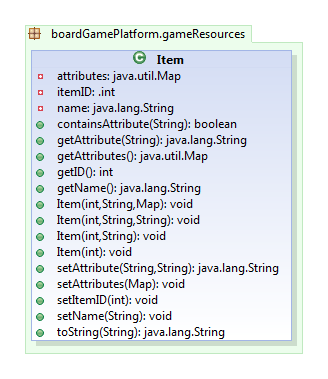
Klasa reprezentująca plansze na której odbywa sie rozgrywka.

1. Field



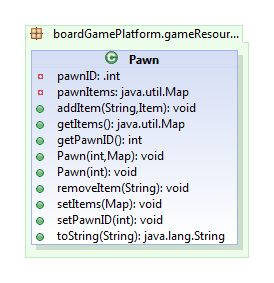
Klasa reprezentująca pojedyncze pole planszy.

1. Item



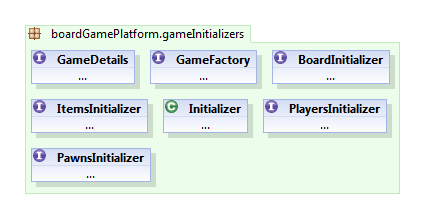
Klasa reprezentująca dodatkowe przedmioty niezbędne do przebiegu rozgrywki.

1. Pawn



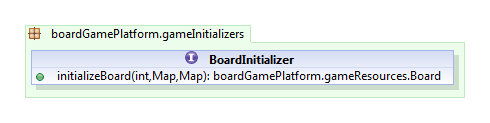
Klasa reprezentująca pionek.

### Pakiet gameInitializers



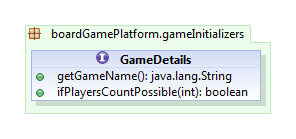
Interfejsy:

1. BoardInitializer



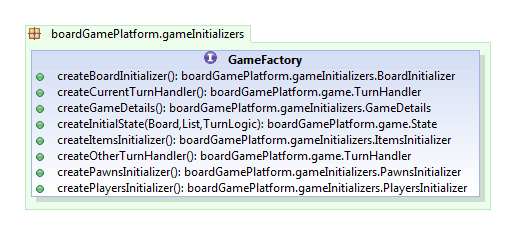
Interfejs umożliwiający inicjalizacje planszy - implementowany przez twórcę gry. Twórca gry, implementujący interfejs jest zobowiązany do określenia początkowego modelu planszy, która będzie wykorzystywana w tworzonej przez niego grze.

1. GameDetails



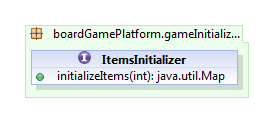
Interfejs umożliwiający określenie szczegółów gry, implementowany przez twórcę gry. Twórca gry, implementujący interfejs jest zobowiązany do określenia nazwy gry oraz ilości graczy dla których możliwe jest przeprowadzenie rozgrywki.

1. GameFactory



Fabryka gry implementowana przez twórcę gry. Twórca gry, implementujący interfejs zobowiązany jest do określenia w jaki sposób zainicjalizowane zostaną niezbędne elementy modelu rozgrywki, takie jak: przedmioty dodatkowe, pionki, plansza, gracze oraz obiekty TurnHandler zawierające informacje o sposobach obsługi tur po stronie graczy. Poza tym twórca gry określa tutaj nazwę gry, oraz ilości graczy dla których możliwe jest przeprowadzenie rozgrywki.

1. ItemsInitializer



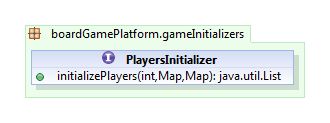
Interfejs umożliwiający inicjalizacje przedmiotów dodatkowych wykorzystywanych w rozgrywce - implementowany przez twórcę gry. Twórca gry, implementujący interfejs jest zobowiązany do określenia przedmiotów dodatkowych które są niezbędne do prawidłowego przebiegu gry.

1. PawnsInitializer



Interfejs umożliwiający inicjalizacje pionków wykorzystywanych w rozgrywce - implementowany przez twórcę gry. Twórca gry, implementujący interfejs jest zobowiązany do określenia pionków, które będą brały udział w grze.

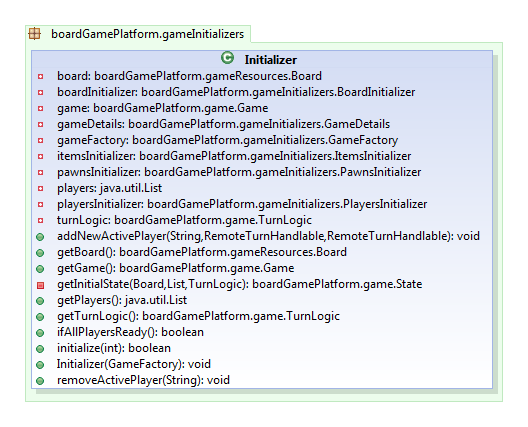
1. PlayersInitializer



Interfejs umożliwiający inicjalizacje listy graczy biorących udział w rozgrywce - implementowany przez twórcę gry. Twórca gry, implementujący interfejs jest zobowiązany do określenia szczegółowych parametrów graczy, którzy będą brali udział w grze.

Klasy:

1. Initializer



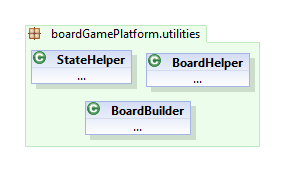
Klasa zajmująca sie inicjalizacja gry, na podstawie zaimplementowanej przez twórcę gry fabryki [GameFactory](file:///C:\\Users\\d00d171\\Desktop\\BoardGamePlatform\\doc\\boardGamePlatform\\gameInitializers\\GameFactory.html" \o "interface in boardGamePlatform.gameInitializers). Dzięki informacjom dostarczonym przez twórcę w implementacji fabryki zostają tu przeprowadzone odpowiednie kroki, w wyniku których uzyskany zostaje gotowy obiekt [Game](file:///C:\\Users\\d00d171\\Desktop\\BoardGamePlatform\\doc\\boardGamePlatform\\game\\Game.html" \o "class in boardGamePlatform.game). Pełny przebieg inicjalizacji wygląda następująco:

1) pierwszym krokiem inicjalizacji (metoda [initialize](file:///C:\\Users\\d00d171\\Desktop\\BoardGamePlatform\\doc\\boardGamePlatform\\gameInitializers\\Initializer.html" \l "initialize(int))) jest stworzenie planszy, listy graczy, oraz ustawienie stanu początkowego. Wykorzystywana jest do tego, podana w konstruktorze klasy, implementacja fabryki GameFactory, dostarczona przez twórcę gry,

2) po pierwszym kroku zainicjalizowana lista graczy nie zawiera jeszcze konkretnego odwzorowania na faktycznych uczestników rozgrywki, ponieważ obiekty [Player](file:///C:\Users\d00d171\Desktop\BoardGamePlatform\doc\boardGamePlatform\game\Player.html) wykorzystywane do reprezentowania graczy nie maja ustawionego nicku, oraz [TurnHandlerow](file:///C:\\Users\\d00d171\\Desktop\\BoardGamePlatform\\doc\\boardGamePlatform\\game\\TurnHandler.html" \o "class in boardGamePlatform.game), które sa niezbedne do poprawnego przeprowadzenia rozgrywki,

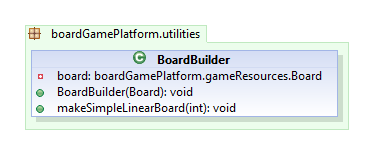
3) w związku z powyższym kolejnym niezbędnym krokiem jest dodanie graczy faktycznie biorących udział w rozgrywce, poprzez przekazanie ich [TurnHandlerow](file:///C:\\Users\\d00d171\\Desktop\\BoardGamePlatform\\doc\\boardGamePlatform\\game\\TurnHandler.html" \o "class in boardGamePlatform.game) oraz nicków.

### Pakiet gameUtilites



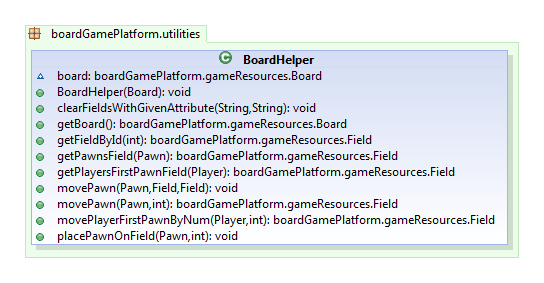
Klasy:

1. BoardBuilder



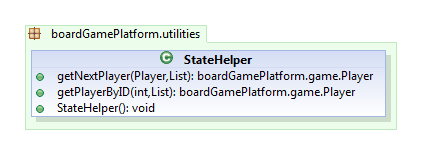
Klasa ułatwiająca tworzenie nowej planszy.

1. BoardHelper



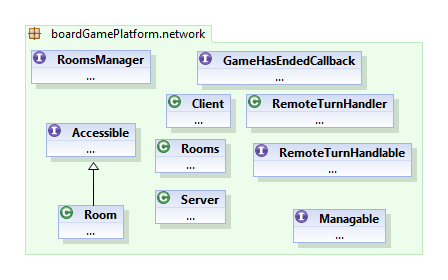
Klasa ułatwiająca dokonywania podstawowych operacji na planszy.

1. StateHelper



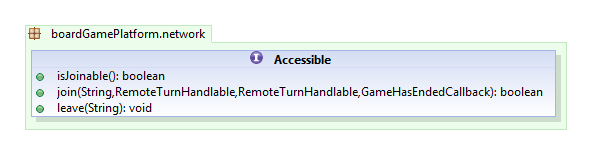
Klasa ułatwiająca operacje na stanach.

### Pakiet network



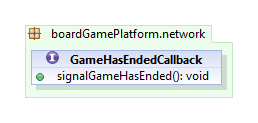
Interfejsy:

1. Accessible



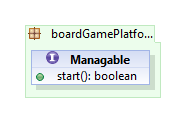
Interfejs umożliwiający dostęp do pokoju. Implementowany przez klasę Room. Daje możliwość dołączenia do niego oraz opuszczenia go. Wykorzystywany jest przez graczy którzy nie są właścicielami pokoju, a chcą tylko wziąć udział w rozgrywce.

1. GameHasEndedCallback



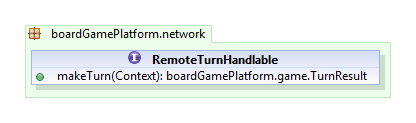
Interfejs używany do zdalnej komunikacji serwera z klientem.

1. Managable



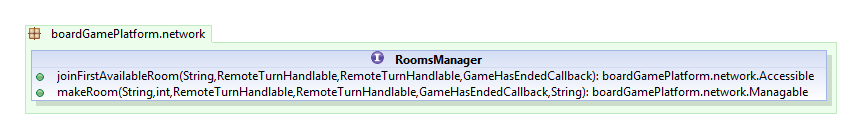
Interfejs umożliwiający zarządzanie pokojem. Implementowany przez klasę Room. Wykorzystywany jest przez graczy którzy są właścicielami pokoju.

1. RemoteTurnHandlable



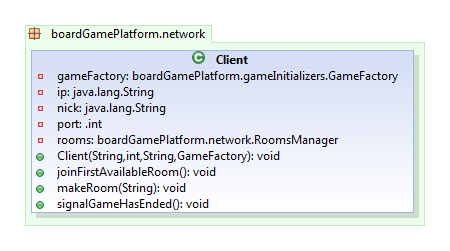
Interfejs wykorzystywany do zdalnego odpytywania graczy biorących udział w rozgrywce o kolejne tury. Implementowany przez RemoteTurnHandler.

1. RoomsManager



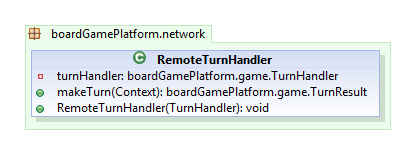
Klasy:

1. Client



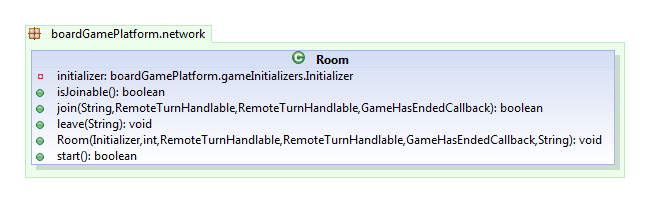
Klasa reprezentująca klienta, łączącego sie z serwerem gry, w celu stworzenia nowego pokoju do gry, lub dołączenia do istniejącego pokoju.

1. RemoteTurnHandler



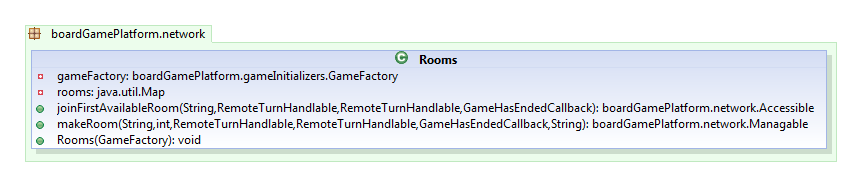
Klasa implementująca RemoteTurnHandlable. Obudowuje ona TurnHandler definiowany przez twórcę gry w momencie określania modelu gry, w celu udostępnienia go serwerowi, aby dać możliwość zdalnego odpytywania klientów.

1. Room



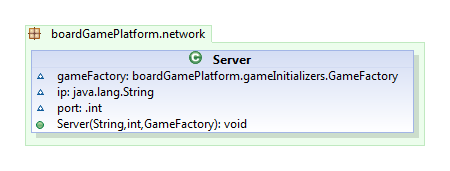
Klasa reprezentująca pojedynczy pokój na serwerze, w którym odbywa sie gra.

1. Rooms



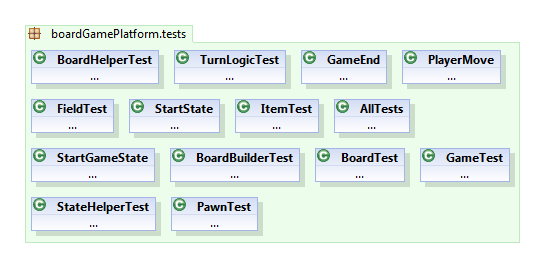
Klasa implementująca RoomsManager. Przechowuje informacje o wszystkich pokojach założonych na serwerze. Umożliwia klientom odpytywanie o nie, oraz dołączanie do wybranego. W momencie tworzenia nowego obiektu określana jest logika gry która będzie obsługiwana przez wszystkie pokoje tworzone na serwerze poprzez, przekazanie w konstruktorze obiektu implementującego GameFactory.

5. Server

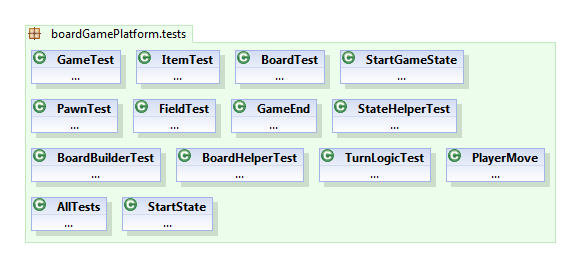


Klasa reprezentująca główny serwer gry. Przechowuje on listę pokoi, w których odbywa sie gra, lub które czekają na dołączenia odpowiedniej liczby graczy. Odpowiada także za udostępnianie listy klientom którzy tego zażądają.

### Pakiet platformExceptions



### Pakiet tests



# Bibliografia

1. Brett D. McLaughlin, Gary Police & David West: *Object Oriented Analysis and Design*, O’Reilly, 2007

Indeks ważniejszych terminów

**Nie można odnaleźć żadnych haseł indeksu.**

1. Spis ilustracji i tabel

Ilustracje

**Nie można odnaleźć pozycji dla spisu ilustracji.**

Tabele