Dokumentacja koncepcji i architektury

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. Analiza problemu 4

2.1. Sformułowanie problemu 4

2.2. Rozpoznanie problemu na przykładach 4

2.3. Analiza problemu 5

2.4. Podział odpowiedzialności 6

3. Architektura systemu 7

3.1. Diagram wdrożenia 7

3.2. Diagram komponentów 8

4. Przepływ sterowania dla podstawowych funkcji platformy 12

4.1. Tworzenie nowego pokoju gry 12

4.2. Dołączanie do istniejącego pokoju 13

4.3. Przebieg gry 14

4.4. Inicjalizacja obiektu gry 15

5. Bibliografia 17

Indeks ważniejszych terminów 18

Spis ilustracji i tabel 18

# Zakres dokumentu

Dokument ten ma na celu przedstawienie wysokopoziomowej architektury platformy do gier planszowych.

Zawiera on szczegółową analizę problemu oraz wysokopoziomową architekturę rozwiązania.

# Analiza problemu

## Sformułowanie problemu

Tematem naszego przedsięwzięcia projektowego jest zaprojektowanie platformy do gier planszowych na system Android. Platforma ma umożliwiać grę dla kilku osób z wykorzystaniem Bluetooth.

Aby przedstawić możliwości platformy zostanie na niej zaimplementowana gra Monopoly.

## Rozpoznanie problemu na przykładach

Aby lepiej zrozumieć istotę naszego problemu postanowiliśmy przeprowadzić analizę typowych gier planszowych, by wyszczególnić podobieństwa i różnice dotyczące planszy, logiki gry, losowości rozgrywki oraz innych istotnych elementów. Poddaliśmy analizie następujące gry:

1) Monopoly

- rozgrywka turowa,

- ruch losowy determinowany przez kostkę,

- pola specjalne, m.in. więzienie, „pola niespodzianki”, pola kupna akcji, pole start,

- pola mają przypisane właściwości: cena, przynależność do grupy,

- w trakcie rozgrywki gracz może modyfikować właściwości pola, poprzez jego zakup, kupno budynków,

2) Grzybobranie

- rozgrywka turowa,

- każdy z graczy otrzymuje na początku koszyczek,

- na odpowiednich polach planszy ustawiane są grzybki – jadalne lub niejadalne,

- jeżeli gracz stanie na polu na którym znajdzie się grzybek jadalny zbiera go do koszyczka,

- jeżeli gracz znajdzie się na polu z grzybkiem niejadalnym traci jeden grzybek,

- jeżeli w koszyczku gracza skończą się grzybki czeka dwie kolejki,

- pola specjalne, m.in. „Stare rozłożyste drzewo napędziło ci strachu” – gracz cofa się o pięć pól, „Zająłeś się zabawą z napotkanym zajączkiem, nie myślisz o zbieraniu grzybów” – gracz traci jedną turę.

3) Pędzące Żółwie – Wyścig do Sałaty

- rozgrywka turowa,

- ruch losowy determinowany przez kostkę,

- kolor gracza jest przydzielany losowo, gracz zna tylko kolor swojego żółwia,

- każdemu z graczy przyporządkowane jest 5 losowo wybranych kart,

- w każdej turze gracz wybiera jedną spośród swoich pięciu kart, które mogą określać ruch jego bądź przeciwników,

- w momencie kiedy na zajęte pole wejdzie inny żółw staje on na żółwiu który już zajmuje pole,

- w momencie ruchu żółw zabiera ze sobą wszystkie żółwie które stanęły na jego grzbiecie,

- wygrywa żółw który jako pierwszy dojdzie do mety,

- w przypadku gdy na mecie znajdzie się kilka żółwi ułożonych jeden na drugim wygrywa żółw znajdujący się na dole

Na podstawie przeanalizowanych przez nas przypadków zdecydowaliśmy się, że nasza platforma będzie wspierać prosty, klasyczny model gier planszowych, w których można wyszczególnić następujące cechy:

- rozgrywka jest turowa,

- występują pola specjalne,

- ruch jest skokowy, nie płynny,

## Analiza problemu

Najważniejszym aspektem, który należy rozważyć w trakcie realizacji projektu jest elastyczność platformy, by umożliwić wspieranie różnego rodzaju gier planszowych, zarówno ze względu na wygląd logikę rozgrywki. W związku z tym ważnymi kwestiami do rozważenia są:

a) plansza

- różne rozmiary (zmienna ilość pól w zależności od gry),

- możliwość definiowania różnych atrybutów pól,

-- różne obiekty na polu

-- tworzenie pól specjalnych

-- tworzenie własnych akcji specjalnych związanych z graczem

b) pionek

- możliwość definiowania różnych atrybutów,

c) logika gry

- możliwość określenia stanu początkowego rozgrywki,

- logika przemieszczania się pionków na planszy

- określanie tur kolejnych graczy

- sprawdzanie stanu rozgrywki

d) grafika gry

- reprezentacja graficzna przystosowana do małych ekranów urządzeń mobilnych,

- możliwość indywidualizacji wyglądu pionka,

- możliwość zmiany wyglądu planszy,

- możliwość dostosowanie wyglądu poszczególnych pól,

- grafika w założeniu 2D (uwzględnienie rozszerzenia do grafiki 3D jest na tym etapie kwestią dyskusyjną).

## Podział odpowiedzialności

Twórca gry jest zobowiązany do:

- zdefiniowania sposobu reprezentacji graficznej poszczególnych elementów gry

- dostarczenie klas stanowiących logikę gry na podstawie modelu

- napisanie mechanizmów odpowiedzialnych za pobranie ruchu od gracza

Platforma jest odpowiedzialna za:

- wyświetlanie grafiki na podstawie zdefiniowanej przez twórcę gry grafiki oraz modelu rozgrywki

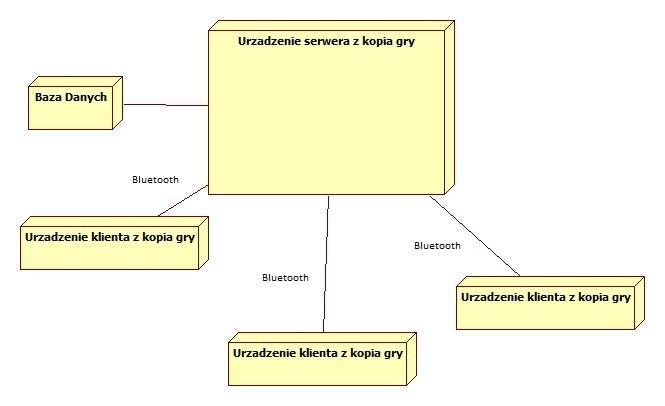
- zapewnienie komunikacji sieciowej

- dostarczenie podstawowych mechanizmów tworzenie gry (klasy odpowiedzialne za

plansze, pola, pionki)

# Architektura systemu

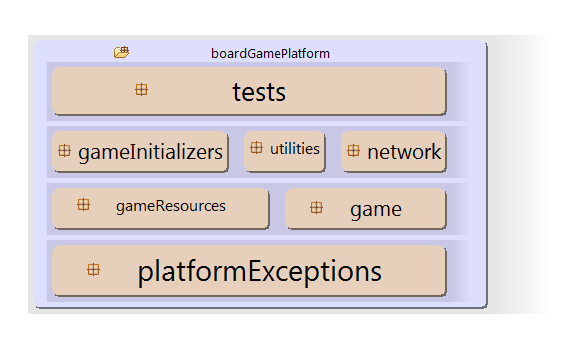
## Diagram wdrożenia



Rysunek ‑ Diagram wdrożenia

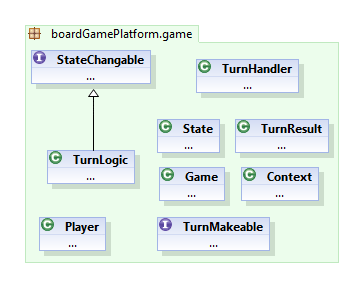
Założyliśmy ze nasza platforma będzie działała w architekturze klient serwer z przesyłaniem wiadomości przez Bluetooth. Osoba która zakłada grę jest serwerem który oczekuje na polaczenia od innych graczy którzy staja sie klientami w naszym modelu. Serwer ma dostęp do bazy danych gdzie można zapisywać i odczytywać stany gry.

## Diagram komponentów



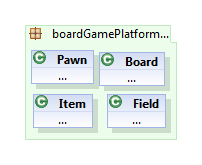
Rysunek ‑ Diagram komponentów

1. game



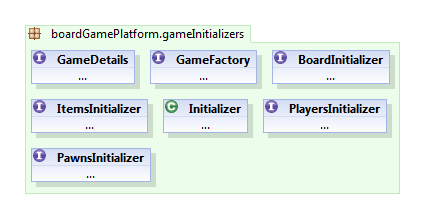
Moduł zawiera wszystkie istotne klasy dotyczące modelu rozgrywki, oraz obsługi jej przebiegu.

2. gameResources



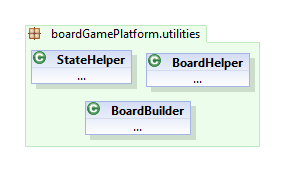
Moduł zawiera wszystkie klasy niezbędne do tworzenia podstawowych elementów rozgrywki, takich jak pionki, plansza, pola planszy, czy dodatkowe elementy gry (np. karty specjalne itp. w postaci itemow).

3. gameInitializers



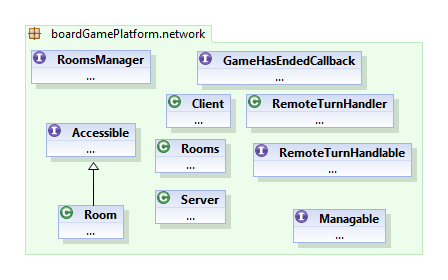
Moduł zawiera klasy niezbędne do inicjalizacji przebiegu rozgrywki w sposób określony przez gracza poprzez implementacje zawartych interfejsów.

4. utilities



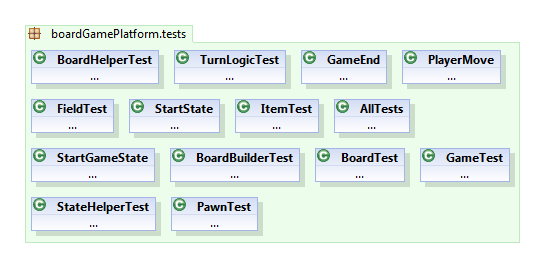
Moduł pomocniczy, ułatwiający korzystanie z niektórych możliwości platformy.

5. network



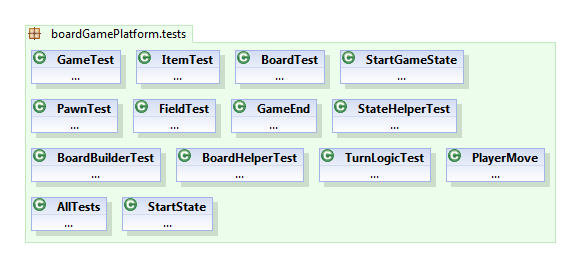
Moduł zawiera mechanizmy do obsługi rozgrywki sieciowej dla wielu graczy podłączonych do gry jednocześnie.

6. platformExceptions



Moduł zawierający wyjątki rzucane przez wszystkie pakiety wchodzące w skład platformy.

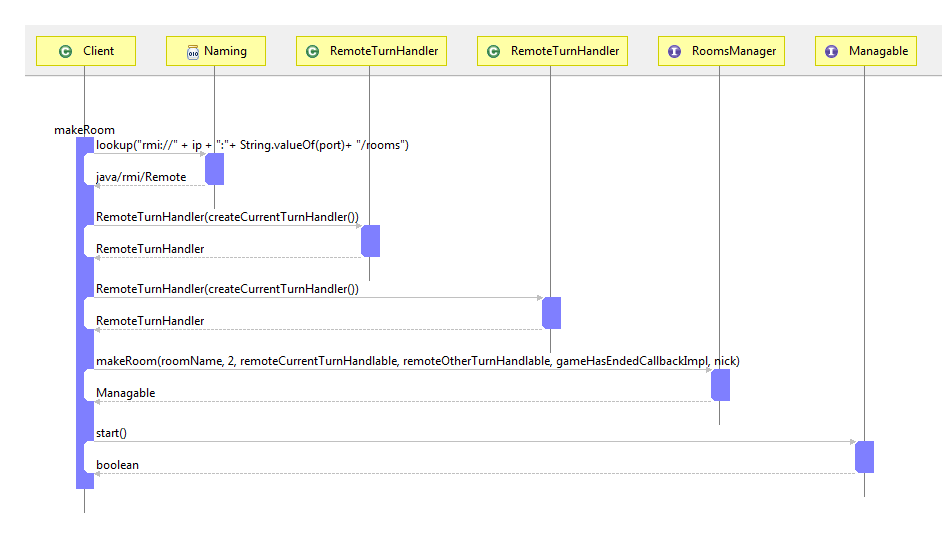
7. tests



Moduł zawiera testy tworzone na potrzeby sprawdzania poprawności działania platformy z pakietów boardGamePlatform.game , boardGamePlatform.gameResources, boardGamePlatform.utilities.

# Przepływ sterowania dla podstawowych funkcji platformy

## Tworzenie nowego pokoju gry

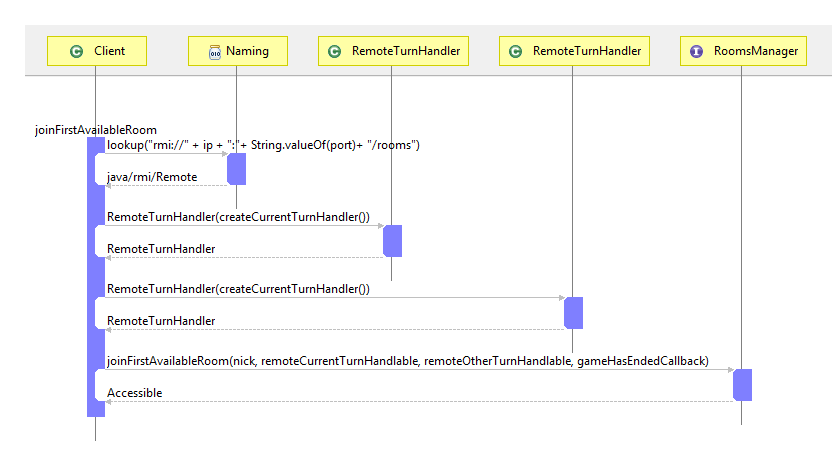


W celu utworzenia pokoju klient odpytuje serwer o listę pokoi. Kiedy ją otrzyma inicjalizowane są obiekty do obsługi tur gracza (zarówno do obsługi gracza w przypadku gdy jest jego tura jak i wtedy gdy jest tura przeciwnika) oraz jego obiekt callback, który zostanie przekazany serwerowi i posłuży do informowania gracza o zakończeniu rozgrywki.

Następnie na obiekcie przechowującym listę pokoi, zwróconym przez serwer, wywoływana jest metoda makeRoom, zwracająca obiekt implementujący interfejs Managable. Pozwala on na rozpoczęcie rozgrywki.

W momencie wywołania na obiekcie przechowującym listę pokoi metody makeRoom klient przekazuje obiekty służące serwerowi do odpytywania go o ruchy w kolejnych turach oraz obiekt callback to informowania o końcu gry.

## Dołączanie do istniejącego pokoju



Aby dołączyć do pokoju klient musi najpierw odpytać serwer o listę istniejących pokoi. Następnie na obiekcie przechowującym listę pokoi, zwróconym przez serwer, wywoływana jest metoda joinFirstAvailableRoom, zwracająca obiekt implementujący interfejs Accessible. Pozwala on na opuszczenie pokoju, jeżeli klient sobie tego zażyczy.

## Przebieg gry

## Przebieg gry.png

W momencie wywołania na obiekcie Game metody start() rozpoczyna się główna pętla programu. W każdej iteracji sprawdzane jest czy gra została zakończona (turnLogic.gameEnded()). Jeżeli nie, obiekt zarządzający logiką gry jest odpytywany o gracza którego obecnie jest tura oraz o obecny kontekst gry.

Następnie przygotowywane są obiekty, które będą przechowywać rezultaty tury. Dla gracza którego obecnie jest tura jest to pojedynczy obiekt TurnResult, natomiast dla pozostałych graczy lista obiektów tej klasy.

W kolejnym kroku gracze są odpytywani o wykonanie tury. W tym celu wykorzystywana jest metoda obiektów RemoteTurnHandler przekazanych przez graczy w momencie dołączania. Zostaje jej przekazany także obecny kontekst gry dzięki czemu wszyscy gracze otrzymują informacje o obecnej sytuacji tj.:

- obecnym stanie planszy,

- liście aktywnych graczy biorących udział w rozgrywce,

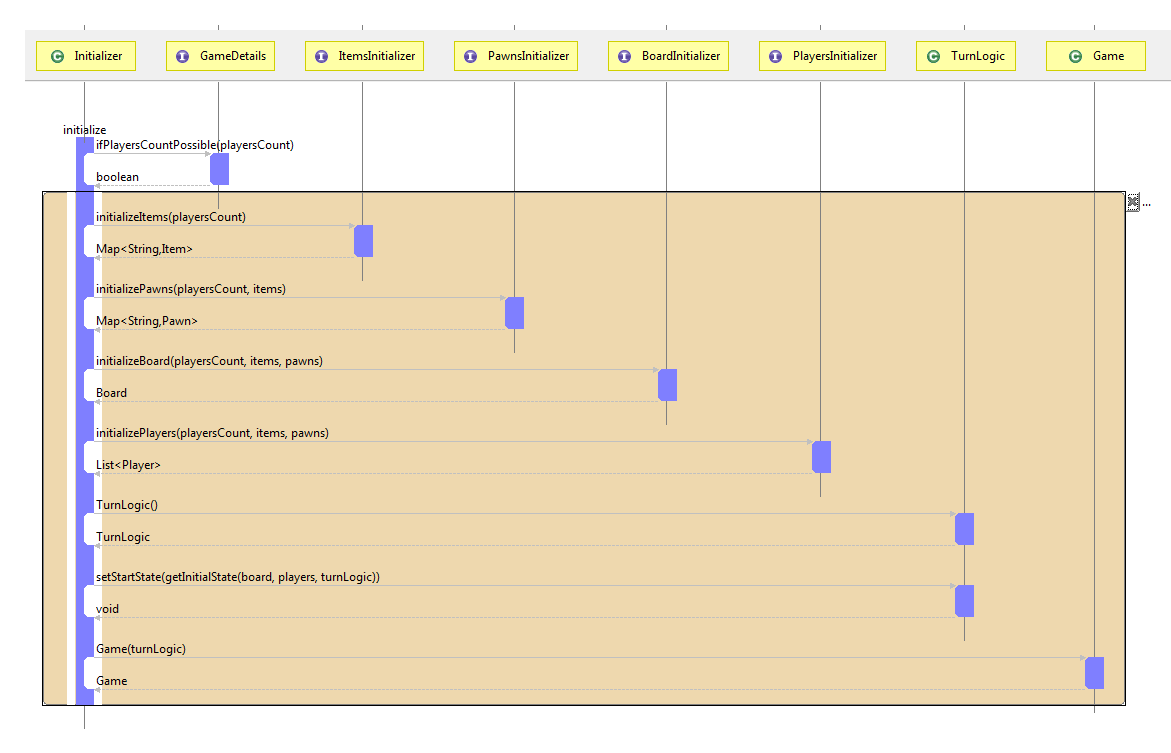
- obecnym stanie gry.

Sposób w jaki obecna sytuacja zostanie zinterpretowana po stronie każdego z klientów (rozróżnienie na gracza którego obecnie jest tura i pozostałych) jest zależny od implementacji gry przez twórcę gry.

Po wykonaniu wymaganych operacji po stronie graczy każdy z nich zwróci wynik tury.

Wyniki te zostaną następnie przekazane do logiki gry, która zinterpretuje je w sposób zdefiniowany przez gracza.

## Inicjalizacja obiektu gry



Za inicjalizację obiektu gry odpowiedzialna jest obiekt klasa Initializer. W momencie tworzenia obiektu tej klasy w konstruktorze przekazywany jest do niej obiekt implementujący GameFactory, implementowany przez twórcę gry, zawierający wszystkie informacje konieczne do inicjalizacji początkowego stanu gry, tzn:

- planszy,

- listy graczy, wraz z ich atrybutami i przedmiotami dodatkowymi,

- pól wraz z ich atrybutami i przedmiotami dodatkowymi,

- dodatkowych przedmiotów koniecznych do prawidłowego przebiegu rozgrywki,

- pionków wraz z ich dodatkowymi atrybutami.

Poza tym obiekt implementujący GameFactory pozwala na uzyskanie informacji o nazwie implementowanej gry, początkowym stanie gry oraz możliwych ilościach graczy, dla których może zostać przeprowadzona rozgrywka.

Dzięki takiemu rozwiązaniu sama inicjalizacja ogranicza się do wykonania prostego algorytmu, wykorzystującego fabrykę dostarczoną przez twórcę. Wykonywane są następujące kroki:

1) sprawdzenie czy możliwe jest zainicjalizowanie rozgrywki dla wymaganej ilości graczy

2) jeżeli nie zwracany jest false

3) jeżeli tak następuje inicjalizacja kolejnych elementów:

- przedmiotów dodatkowych

- pionków

- planszy

- graczy

- logiki gry, z określonym przez twórcę gry stanem początkowym

- samego obiektu gry.

# 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

Rysunek 2‑3 Diagram wdrożenia 8

Rysunek 2‑4 Diagram komponentów 9

Tabele