Technologie programowania - 2024
Lista 6

Trylma (ang. chinese checkers)

Celem jest zaprojektowanie i implementacja systemu do prowadzenia rozgrywek w chińskie warcaby. System należy tworzyć **w parach wykorzystując repozytorium**. Historia zmian w repozytorium będzie podstawą do oceny wkładu członków zespołu.

Uwaga: Robimy iterację 3 o funkcjonalności podanej poniżej. Program powinien więc posiadać funkcjonalności wersji 1, 2 oraz 3. Wymagany jest osobny link na github dla każdej iteracji.

Zasady gry

Zasady są dosyć łatwe, więc poniżej przytoczę je jedynie pobieżnie. Opis bardziej szczegółowy znajduje się na wiki oraz tutaj.

1. W jednej grze uczestniczy 2, 3, 4 lub 6 graczy. Każdy gracz ma 10 pionów.
2. Gra toczy się na planszy w kształcie sześciopromiennej gwiazdy. Każdy promień zawiera 10 pól, w których początkowo umieszczane są piony. Wewnętrzny sześciokąt ma 61 pól, na każdy jego bok składa się 5 pól.
3. Celem każdego gracza jest umieszczenie wszystkich swoich pionów w przeciwległym promieniu. Wygrywa gracz, który dokona tego jako pierwszy. Pozostali gracze mogą kontynuować grę walcząc o kolejne miejsca.
4. Gracz rozpoczynający grę wybierany jest losowo. Następnie gracze wykonują ruchy po kolei zgodnie z kierunkiem ruchu wskazówek zegara.
5. Ruch polega na przesunięciu piona na sąsiednie puste pole lub na przeskakiwaniu pionem innych pionów (swoich lub przeciwnika) tak jak w warcabach, ale bez możliwości zbijania pionów. Ruch może odbywać się w dowolnym kierunku, poza przypadkiem, gdy pion wejdzie do docelowego (przeciwległego) promienia. Wówczas pion może wykonywać ruchy jedynie w obrębie tego promienia.
6. Dopuszczalne jest pozostawianie swoich pionów w promieniu docelowym innego gracza (tzw. blokowanie).
7. Gracz może zrezygnować z ruchu w danej turze.

Wymagania funkcjonalne dotyczące iteracji 1.

W iteracji 1 należy zaimplementować następujące wymagania funkcjonalne (**tylko takie !!**):

1. System powinien działać w oparciu o architekturę klient-serwer. W razie wątpliwości warto przyjrzeć się przykładom z tej strony.
2. Gracz za pomocą aplikacji klienckiej powinien móc połączyć się z serwerem i dołączyć do gry.
3. System powienien dać możliwość wprowadzenia ilości graczy według reguł opisanych w punkcie powyżej.

4. Klient powinien móc wysłać ruch do innych klientów. W tej wersji interfejs użytkownika powinien być konsolowy. Z poziomu klienta (na konsoli) należy wpisać pozycję początkową oraz końcową ruchu, które będą wysyłane przez serwer do innych klientów. Do tego celu potrzebna będzie klasa przechowująca planszę np. **Board**.

Wymagania dotyczące iteracji 2.

W iteracji 2 należy zaimplementować następujące wymagania funkcjonalne (**tylko takie !!**):

1. Należy zaimplementować możliwość rozgrywania gry. Należy zaimplementować co najmniej dwie strategie opisane tutaj.
2. Należy stworzyć interfejs okienkowy dla klientów (np. JavaFX).
3. Poprawność kodu powinna być na bieżąco weryfikowana testami jednostkowymi napisanymi przy użyciu JUnit.
4. Należy stworzyć diagram klas oraz ewentualnie inny diagram UML obrazujący funkcjonalność aplikacji.
5. Należy wygenerować dokumentację w JavaDoc.

Wymagania dotyczące iteracji 3.

W iteracji 3 należy zaimplementować następujące wymagania funkcjonalne (**tylko takie !!**):

1. Do gry mogą zostać dołączeni gracze-boty działający na serwerze. Stwórz jedną przykładową implementację bota, która będzie wykonywała choć w miarę sensowne ruchy.
2. Należy rozszerzyć chińskie warcaby o zapisywanie gry do bazy danych używając frameworka **Spring**. Baza danych powinna być w stanie rejestrować poszczególne gry oraz ruchy w poszczególnych grach. Dodatkowo aplikacja powinna umięć odtwarzać daną grę na podstawie zapisu do bazy danych

Dodatkowe wymagania

1. Postaraj się tworzyć system starannie i w przemyślany sposób. Wykorzystaj UML by zaproponować i rozwijać projekt systemu. Systemy chaotyczne zaprojektowane i pisane na ostatnią chwilę będą nisko ocenione. Bierz również pod uwagę, że to co stworzysz będzie miało wpływ na kolejną listę.
2. Stosuj podejście iteracyjne i przyrostowe.
3. Istnieje bardzo wiele wariantów gry, np. inne mogą być rozmiary planszy, inna liczba pionów, inne zasady dotyczące możliwych ruchów i blokowania. **Projektuj system tak by można go było łatwo rozszerzyć o nowe warianty gry.**
4. Postaraj się wykorzystać poznane narzędzia i wzorce. Im więcej ich zastosujesz, tym wyżej będzie ocenione Twoje rozwiązanie. Ich użycie powinno być jednak zawsze uzasadnione.

Punktacja

Za implementację tej części każdy student może dostać maksymalnie **100 punktów**.

Sposób zaliczenia

Pokazanie rozwiązania prowadzącemu oraz wrzucenie na ePortal: linku do githuba zawierającego wszystkie pliki wraz z instrukcją uruchomienia projektu.