Wielki Turniej - SKJ Proj 01

Jakub Parteka

Wszystkie metody są również opisane w kodzie programu, niektóre komentarze w kodzie i tym opisie powtarzają się.

1.Ogólny opis rozwiązania:

Przykładowe dane do uruchomienia klientów w grze:

Karol 7000

Pawel 3000 numerIP(który wyświetli się po uruchomieniu pierwszego gracza) 7000

Kuba 4500 numerIP(który wyświetli się po uruchomieniu pierwszego gracza) 3000

Filip 3200 numerIP(który wyświetli się po uruchomieniu pierwszego gracza) 7000

Marek 6400 **numerIP(który wyświetli się po uruchomieniu pierwszego gracza) 3200**

Pierwszy gracz inicjalizowany jest 2 argumentami przez co **„**skazany**”** jest na oczekiwanie na **następnych graczy, aby to zrobić musi wpisać dowolny znak, do wyboru ma opcje QUIT, która** oczywiście **kończy dla niego rozgrywke. Jednakże żeby program zadziałał, pierwszy klient musi mieć 2 argumenty. Każdy następny uruchamiany jest z 4 argumentami (Imie, jego port, IP osoby z którą będzie się łączył oraz jej port). Dołączający od razu jest łączony z** graczami w grze i rozgrywa z nimi **losową grę w marynarza(kliencie wysyłają między sobą losowe liczby, odliczanie odbywa się po stronie słuchającego klienta). Po zagraniu z wszystkimi graczami klientowi pozostają dwie opcje do** wyboru, QUIT lub wpisanie dow**olnego znaku i czekanie aż ktoś nowy dołączy do gry i się z nim połączy. W przypadku gdy jakiegoś gracza nie ma już w grze, a ktoś próbuje się z nim połączyć to** dostaje on 2 kolejne szanse na **„**odebranie**”** **połączenia. Jeśli nie uda się połączyć gracz jest us**uwany z listy aktywnych graczy.

W programie użyto klas z pakietu java.net (Socket,ServerSocket,InetAdress), Map oraz Scannera.

2.Szczegółowy opis rozwiązania:

a)**Dane na temat graczy są przechowywane między innymi w klasie IPPort(przechowuje ip oraz port).**

Każdy jest identyfikowalny po przez swoją nazwę, IP oraz numer portu.

Dane na temat aktywnych graczy przetrzymywane są w HashMapie(z kluczem IPPort,który implementuje interface Serializable, z nadpisanymi metodami hashcode oraz equals,by można było te obie**kty rozróżnić oraz value -booleanem, który daje informacje czy już rozegraliśmy z danym graczem grę), którą każdy gracz otrzymuje po dołączeniu do rozgrywki. Wyniki również przetrzymywane są w HashMapie(z kluczem String**-Nazwa gracza oraz value- wynik gry)**. Każdy z graczy posiada te dwie mapy, które są dla niego unikalne.**

b) 1.Metoda join () **laczy nas z podanym jako argumenty klientem (próbuje się połączyć 3 razy, jeśli się nie uda daje komunikat że gracz wyszedł i usuwa go z listy graczy)**, pobiera od niego mape graczy, zmienia wartosci graczy w mapie na false (jeszcze nie rozegral z nimi gry).Wywoluje metode play().

2.Metoda play () **zmienia wartosc socket na nowe(jesli port się rozni). Laczy nas z nasluchujacym agentem(próbuje się połączyć 3 razy, jeśli się nie uda daje komunikat że gracz wyszedł i usuwa go z** listy graczy), pobiera od niego mape graczy oraz jego port,wysyla mu losowa liczbe, nazwe naszego gracza. Po czym dostaje nazwe przeciwnika oraz wynik rozegranej gry, wrzuca go do mapy scores oraz wrzuca dane gracza do mapy player z **parametrem true (ponieważ właśnie rozegrał z nim gre). Na koniec uruchamia się rekurencyjnie dla wszystkich graczy z mapy players z ktorymi** jeszcze nie zagral oraz zamyka socket i potoki.

3.**Agent rozegrał rozgrywkę dla wszystkich pozostałych graczy w sieci. Uruchamiana jest metoda** listen()

4.**Metoda listen (), przy każdym obrocie petli pyta uzytownika czy chce nadal grac, jesli nie to** przerywa petle oraz drukuje wyniki wszystkich dotychczasowych gier,jesli chce grac to gracz przechodzi w tryb nasluchiwania polaczen. Po otrzymaniu polaczenia klient,pobiera ip, wysyla swoja **mape graczy nowo połączonemu klientowi, potem klient nastepujaco:**

0.Pobiera port gracza
1.Losuje swoją liczbe.
2.Odbiera od gracza liczbe.
3.Odbiera od gra**cza jego nazwę.**
4.Wysyla mu swoja nazwe.
5.Przeprowadza odliczanie.
6.Zapisuje wynik w mapie scores(nazwa gracza,wynik)
7.Wysyla wynik swojemu przeciwnikowi.
8. Wrzuca przeciwnika do mapy players z oznaczeniem,ze rozegral z nim gre.
9. Zamyka wszystkie potoki oraz polaczenie
10.Zaczyna pętle od nowa

c) **Gracz nie może oszukiwać bez ingerencji w kod, po stronie nasłuchującej zapewnione są 2 źródła losowości (losowanie liczby gracza oraz losowanie od kogo zostanie zaczęte odliczanie)**po stronie **gracza joinującej jest jedno źródło losowości (losowanie liczby gracza).**

Obserwacje, eksperymenty i wnioski

Moje pierwsze podejście do tego projektu niestety skończyło się porażką, ponieważ starałem się je **realizować za pomocą wątków, które spowodowały mase różnych błędów wyskakujących w niespodziewanych momentach, których nie byłem w stanie naprawić. Strumienie binarne również sprawiły mi niemały problem, aczkolwiek po spędzeniu dłuższego czasu czytając dokumentację wszystko stało się jasne. Nie udało mi się zaimplementować monitora http oraz niestarczyło mi czasu na zaimplementowanie porządnego narzędzia uczciwości (np. kryptografii), stąd posłużyłem się zdecydowanie prostszym (jak i mniej efektywnym) rozwiązaniem, mianowicie kilkukrotnymi** e**lementami losowści.**