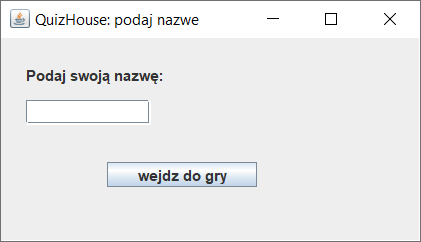
|  |  |
| --- | --- |
| Projekt Java  Politechnika Świętokrzyska | |
| **QuizHouse** | |
| Patryk Banaś  Damian Banasik | 02.06.2020 |
| 2ID11A  2019/2020 | |

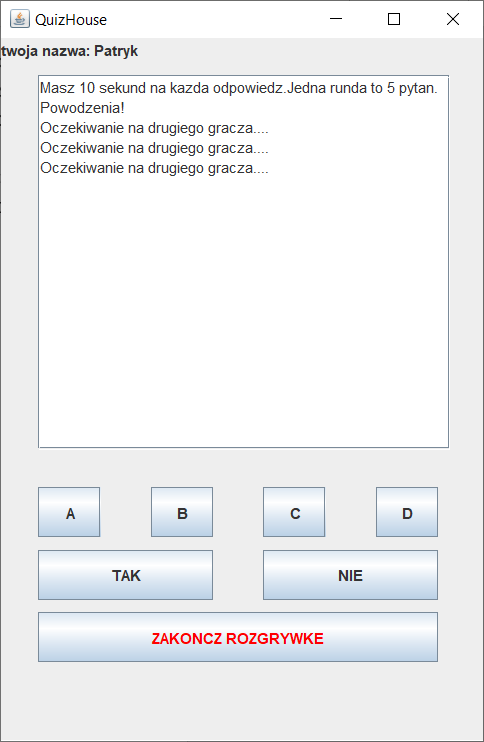
**Opis gry**

Jako temat projektu utworzyliśmy grę pod tytułem QuizHouse. Do jej stworzenia użyliśmy architektury klient-serwer. Gra polega na tym aby 2 graczy spośród wybranej kategorii pytań odpowiadało na pytania, na każde z nich mamy 10s, za każde jest dodawany do naszego konta jeden punkt (serwer zapisuje wynik w swojej klasie). Jedna runda to 5 pytań, po skończonej rundzie serwer ogłasza werdykt oraz daję możliwość ponownej rozgrywki.

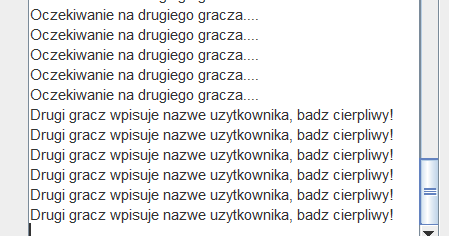
Stworzyliśmy graficzny interfejs dla użytkownika, który wymaga wprowadzenia nazwy gracza, jeśli tego nie zrobimy nie będziemy mogli połączyć się z naszym przeciwnikiem. Serwer nie posiada interfejsu graficznego, ponieważ jest on całkowicie zbędny. Serwer umożliwia połączenia ze sobą tylko 2 graczy. Gra była testowana w sieci lokalnej (działa bez zarzutu). Aby ją uruchomić wystarczy odpalić serwer oraz stworzony program z rozszerzeniem jar. Dołączony do programu. (jest to nasz klient)



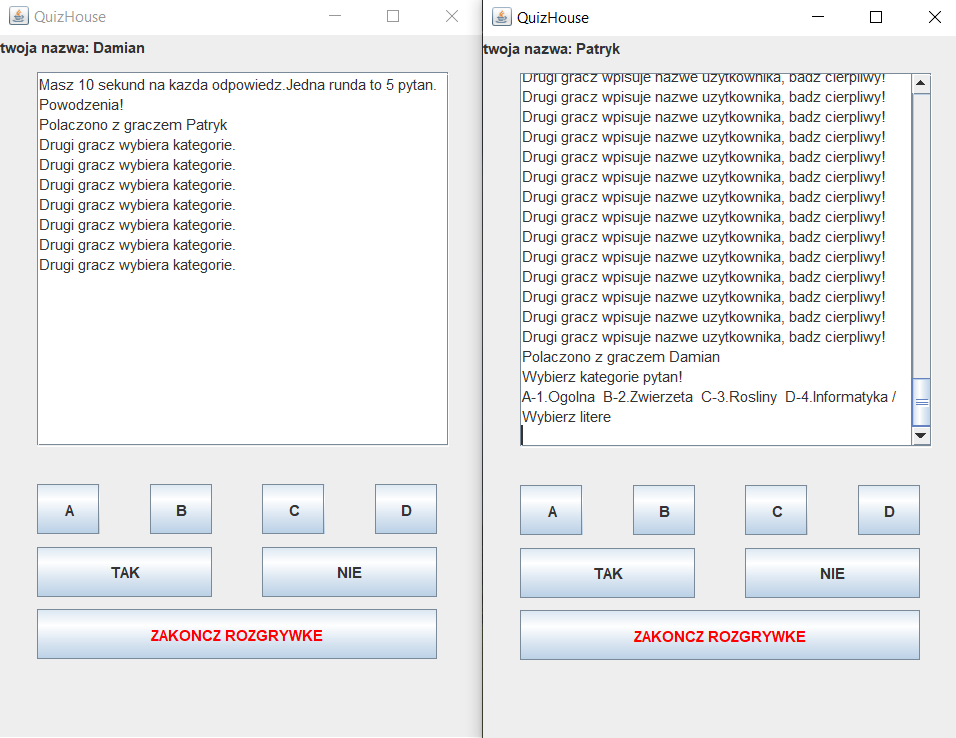


Po jej uruchomieniu podajemy swoją nazwę użytkownika. Program jest zabezpieczony w taki sposób, w przypadku zostawionego pustego pola nie pozwoliło nam przejść dalej. 

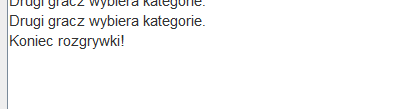
Po wpisaniu nazwy użytkownika przechodzimy do naszego głównego okienka gry. Posiadamy okienko które wyświetla nam komunikaty uzyskane od serwera, oraz kilka przycisków funkcyjnych. Komunikat oczekiwanie na drugiego gracza otrzymywać będziemy tak długo, aż np. z innego komputera nie uruchomi aplikacji klienta. Możliwe jest także uruchomienia 2 programów na jednym komputerze.



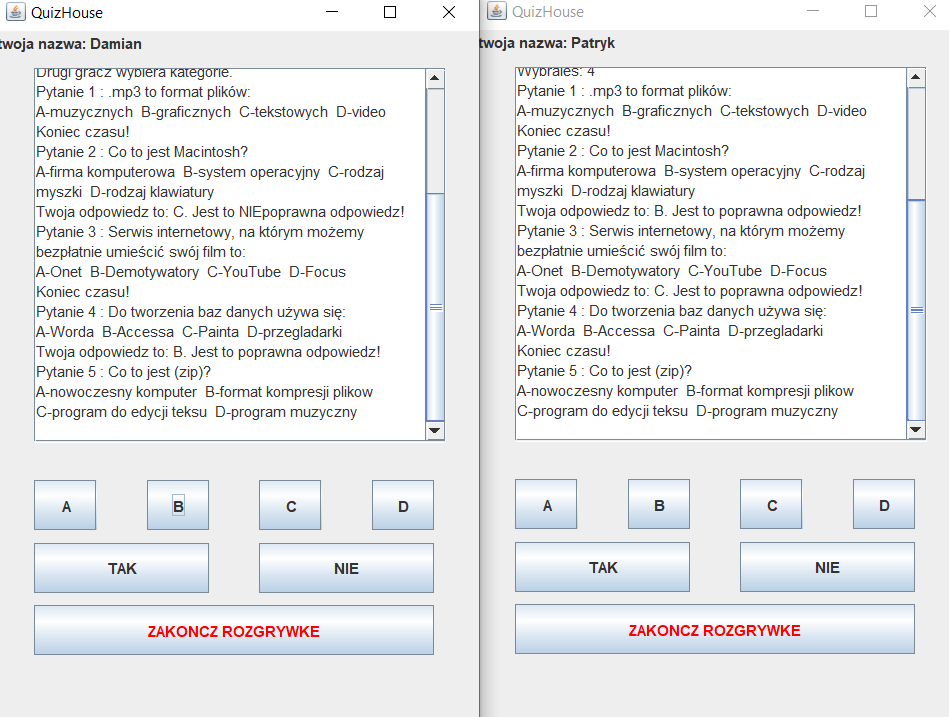
Kiedy 2 program zostanie uruchomiony, komunikat uzyskiwany od serwera się zmieni.



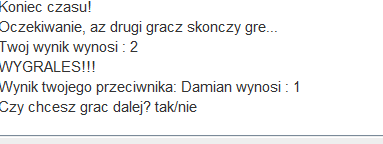
Po podaniu nazwy użytkownika przez drugiego gracza, serwer nas połączy ze sobą, oraz wyświetli nazwę gracza z którym zostaliśmy połączeni. Następnie umożliwi wybór kategorii pytań. Stworzony mechanizm umożliwia graczom sprawiedliwy podział wyboru kategorii, czyli kategorie są wybierane raz przez jednego gracza, a raz przez drugiego.



Gra jest zabezpieczona różnymi licznikami czasowymi, dlatego trzeba się spieszyć w podejmowaniu decyzji. Uznaliśmy że to będzie dobrym rozwiązaniem w przypadku, aby jeden z graczy nie tracił zbyt wiele czasu oczekując na drugiego.

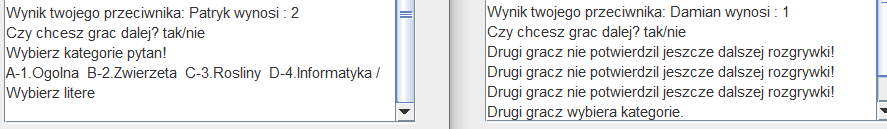


Po wybraniu kategorii przez gracza zaczyna się rozgrywka, serwer wysyła pytania do graczy, oraz odbiera odpowiedzi. Wyświetlane jest również powiadomienie, jaką odpowiedź udzieliliśmy oraz czy jest ona poprawna. W przypadku kiedy w ciągu 10 s nie zdążymy udzielić odpowiedzi wyskakuje komunikat że czas nam się skończył.



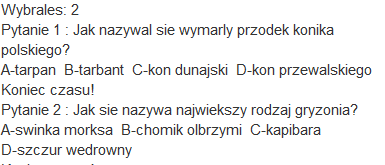
Po skończonej rundzie Serwer wysyła werdykt, naszą liczę punktów oraz liczbę punktów naszego przeciwnika, oraz pyta czy chcemy kontynuować naszą rozgrywkę. Serwer jest również zabezpieczony tym aby nie zakończył rozgrywki przed podsumowaniem punktów, dlatego u gracza który pierwsza zaczął grę, może pojawić się komunikat taki jak „ Oczekiwanie, az drugi gracz skończy gre..”

Oczywiście serwer w tym wypadku również jest zabezpieczony np. tym kiedy gracz wyrazi ochotę na dalszą grę a drugi nie. W takim wypadku wyświetli się komunikat o końcu rozgrywki.



Gdy gracz zaakceptuję dalszą rozgrywkę, serwer wysyła mu komunikaty o działaniach przeciwnika.

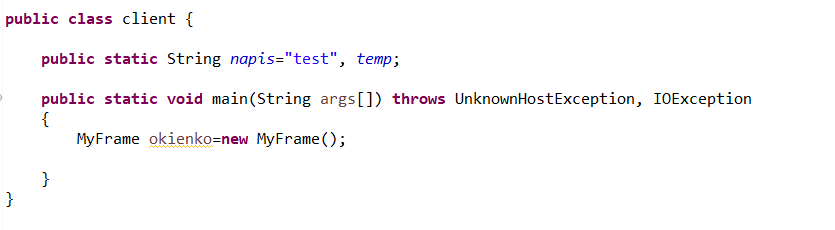
Tym razem kategorię wybiera gracz który poprzednio tego nie robił.



**Opis kodu**

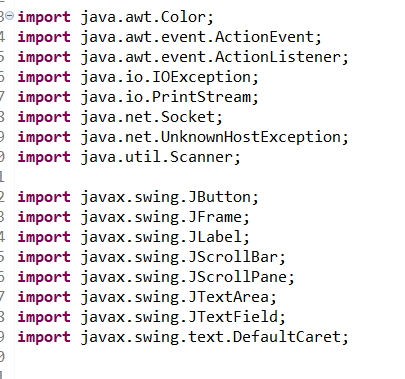
Nasz program jest podzielny na 2 części: klienta i serwer.

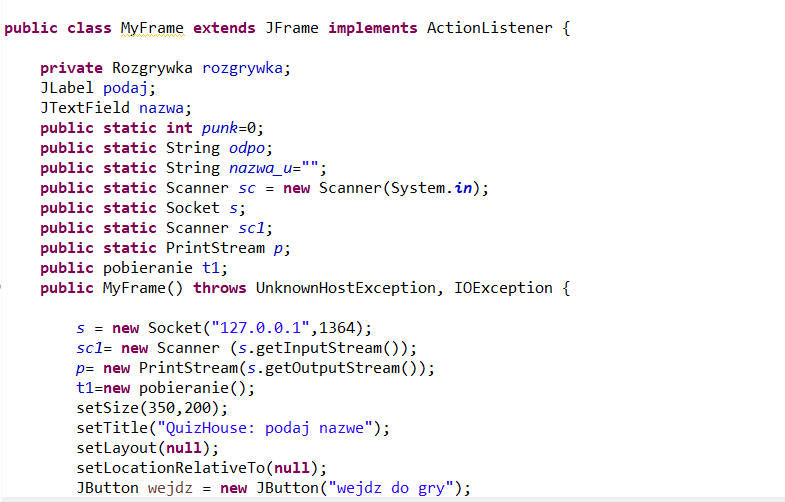
Zacznijmy od opisu klienta.



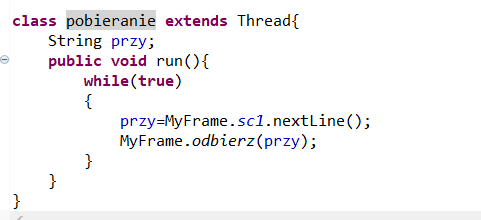
Główna klasa klienta zawiera mało kodu, jest tworzony tutaj obiekt klasy MyFrame, w której jest tworzony cały szkielet naszego graficznego okienka dla gracza, oraz funkcję zapewniające odbierania oraz nadawanie danych do serwera.

Teraz trochę o klasię MyFrame. Konieczne jest zaimportowanie poniższych klas do poprawnego działania programu.

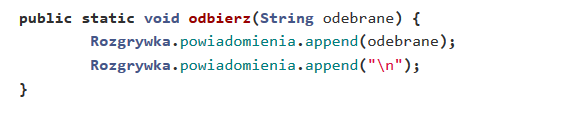




Nasza klasa musi dziedziczyć po klasie JFrame oraz zaimplementować interfejs ActionListener odpowiedzialnego za odróżnianie który przycisk został naciśnięty przez naszego gracza. Rozgrywka to również klasa która jest odpowiedzialna za następne okienko po podaniu nazwy użytkownika. Tworzone jest kilka nowych rzeczy na początku. Np. JLabel to oczywiście przycisk, JTextField to nasze pole w którym będzie wpisywana nazwa gracza pobieranie to klasa tworzący nowy wątek. W tym wypadku adres ip jest ustawiony na pętlę zwrotną oraz port 1364. W przypadku sieci LAN wystarczyłoby tutaj podać adres postawionego serwera z sieci lokalnej.



Klient potrzebuje wątków aby jednocześnie pobierać i wysyłać dane od serwera.

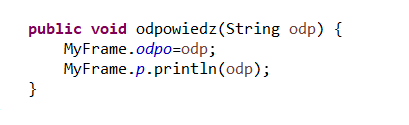


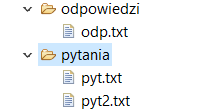
Odbierz to natomiast funkcja odpowiedzialna za wysyłanie odebranych Stringów na ekran.

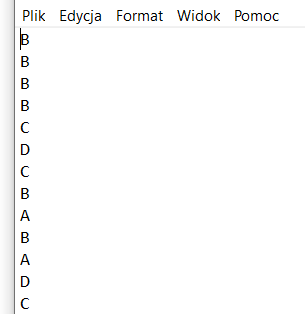




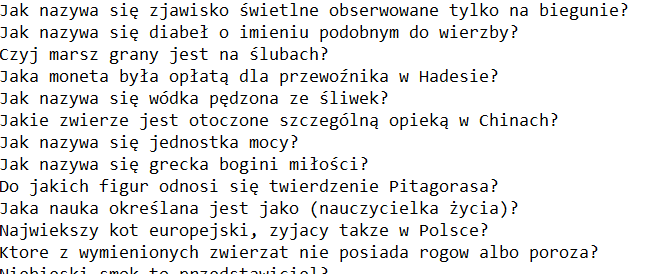
Przyciski są skonstruowane w taki sposób aby wysyłały gotowe teksty do serwera, ten natomiast sprawdza jaką odpowiedź otrzymał od klienta i w zależności od odebranego łańcucha odsyła odpowiedź albo kończy swoje działanie. Dane są wysyłane za pomocą funkcji odpowiedz.

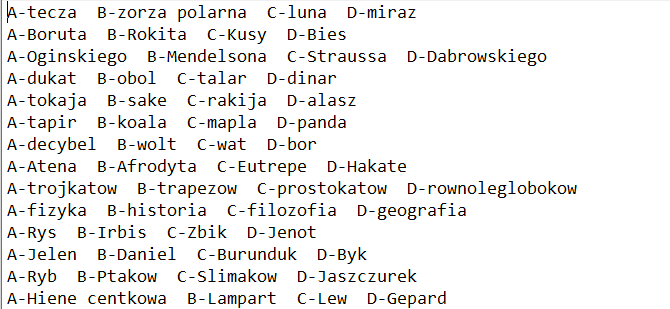






Te 3 pliki tekstowe są odpowiedzialne za pobór pytań, możliwych odpowiedzi oraz ich kluczy, te znajdują się w odp.txt są wyświetlone powyżej, natomiast poniżej jest zawartość plików pyt.txt oraz pyt2.txt.

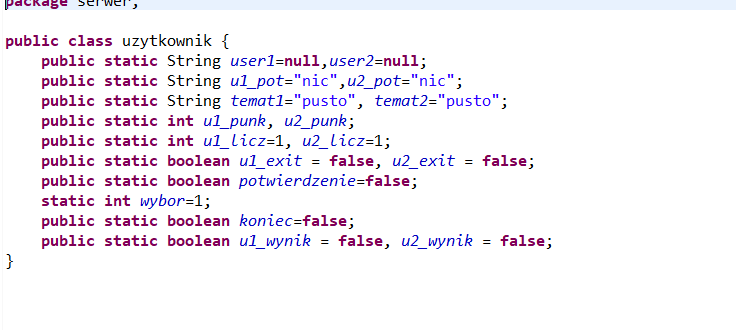




Teraz trochę o działaniu serwera.

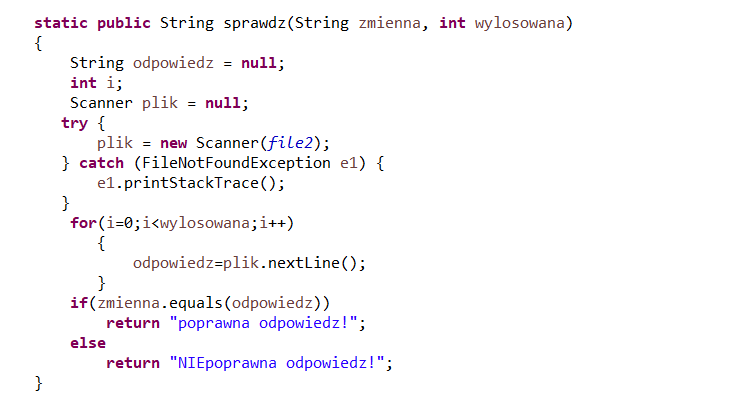


Serwer tworzy gniazdo, używa portu 1364, ważne jest aby klient używał tego samego portu co serwer. Konieczne jest także aby serwer pracował na wątkach, aby 2 klientów mogło rywalizować ze sobą. Ten serwer jest przeznaczony na działaniu tylko dla 2 klientów. Ziarno to natomiast liczba wysyłana do funkcji losującej. Jest ona stworzona aby klienci mieli wspólne pytania.

Można powiedzieć że klasa użytkownik to trochę taka baza danych, przechowuje ona informację np. takie jak nazwa użytkowników, liczbę punktów, czy zmienne odpowiedzialne za to aby serwer wiedział co się w danej chwili dzieję i mógł zabezpieczać pewne sytuację.

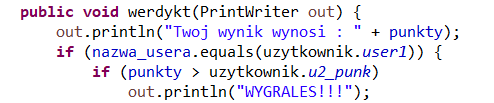


Klasa losowanie jest odpowiedzialna za losowanie pytań, działa ona w taki sposób że zwraca wylosowaną liczbę w zależności jaki temat został wybrany. Ten numer będzie naszą linijką w pliku tekstowym którą odeślemy jako pytanie dla gracza. Funkcja odpowiedzialna za losowanie, zabezpiecza również to aby w danej rundzie nie pojawiły się te same pytania.



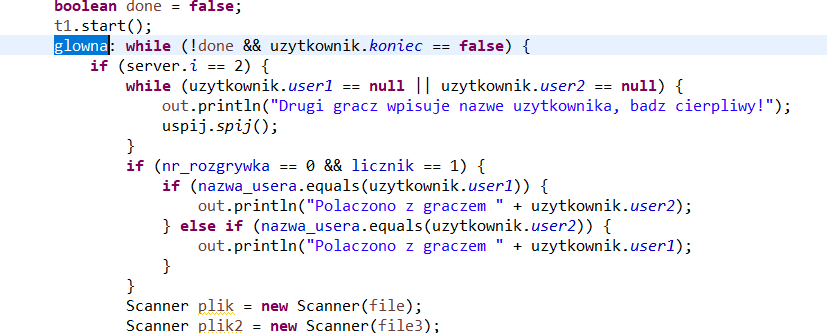
Ta klasa posiada także funkcję sprawdzającą klucze odpowiedzi, zwraca odpowiedni komunikat.

Teraz przyszła pora na omówienia naszej głównej funkcji na której opiera się działanie serwera czyli „ThreadedEchoHandler”,



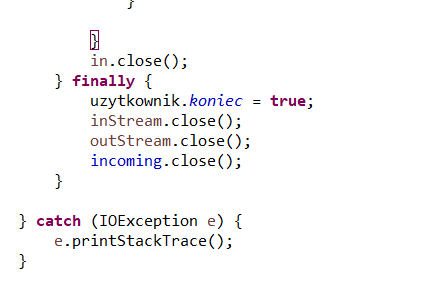
Funkcja werdykt odpowiedzialna za wyświetlanie werdyktu.

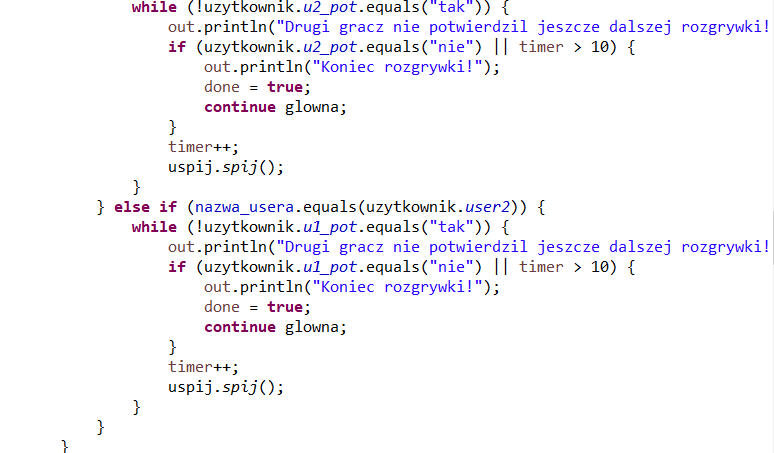
Nasz serwer posiada wiele zabezpieczeń między innymi sprawdza czy nasz przeciwnik dalej jest w rozgrywce, jeśli nie to gra przestanie być kontunuowana.



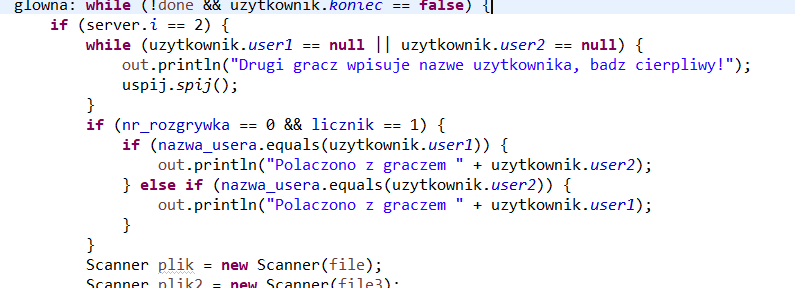
Użyliśmy do tego etykiety

koniec pętli while która kończy się jeśli warunki nie są spełnione bo np. jeden z graczy zakończył już rozgrywkę.:

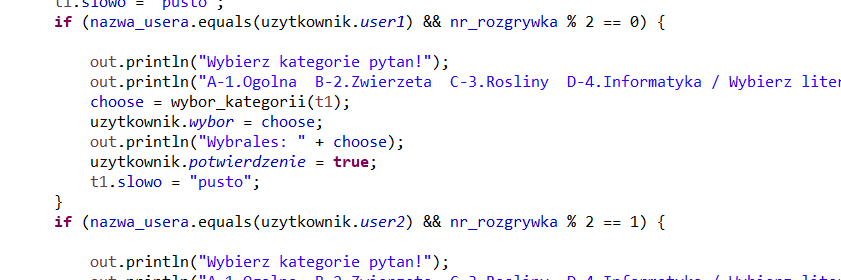




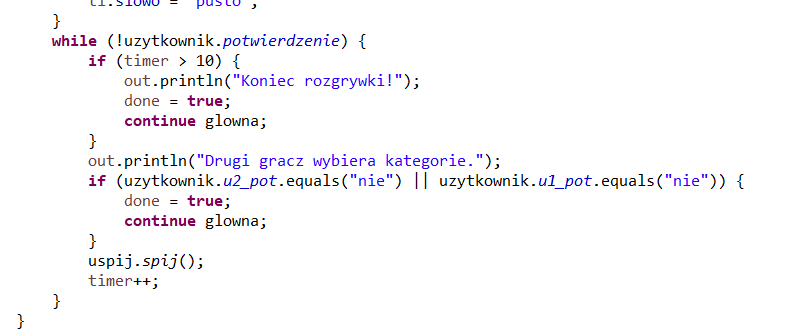
Warunek który przechodzi do etykiety, jeżeli gracz nie wyraził dalszej ochoty na rozgrywkę i nacisnął przycisk „nie” przechodzi do etykiety i warunek pętli nie jest spełniony. Ewentualnie jeśli skończy się czas do namysłu.



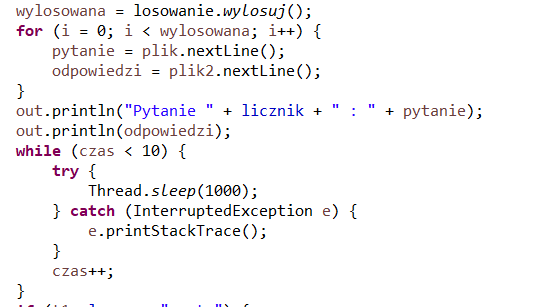
Tutaj natomiast widzimy pętle która pokazuje na ekranie co 2 s komunikat u klienta. Oraz niżej komunikat z jakim graczem zostaliśmy połączeni, właśnie tutaj przyda nam się zmienna nr\_ruzgrywka, ponieważ ten komunikat wykona się tylko raz w trakcie trwania gry, tylko raz licznik jest równy 1 i tylko raz rozgrywka jest równa 0. Licznik to zmienna która inkrementuje się po każdym pytaniu, na koniec rundy zostaje przywrócona domyśla wartość.



Tutaj serwer sprawdza po nazwie gracza oraz numerze rozgrywki kogo kolej jest na wybór kategorii.



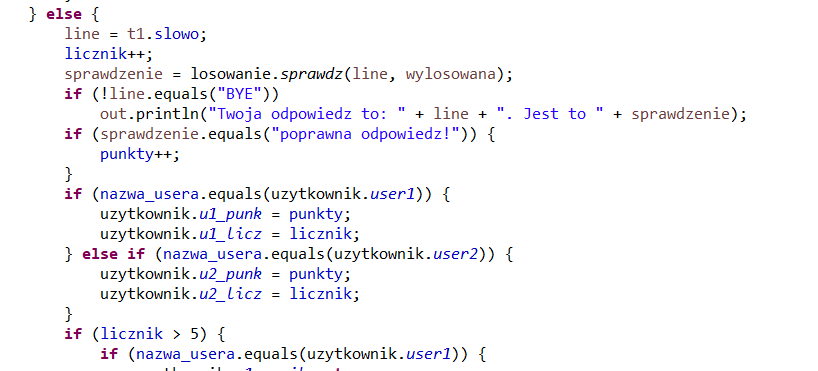
Tutaj następne zabezpieczanie które kończy rozgrywkę w przypadku kiedy np. ktoś się rozmyśli już w trakcie wybierania kategorii. Albo jeśli jeden z graczy robi to zbyt długo.



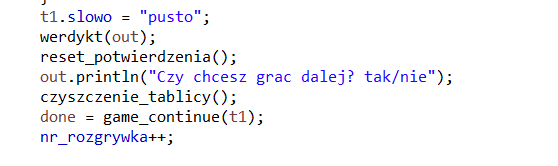
Losowanie pytania, oraz pętla for, mechanizm losowania jest prosty W bazie mamy 40 pytań co 10 zmieniają się kategorie, jeśli gracz wybierze kategorię 1, to losuje się od 1-10 jeśli w to od 11-20 itd. Zwracany jest w ten sposób numer po którym potem dana linijka będzie wyświetlana. Oczywiście można przerabiać liczbę pytań czy losowanie do indywidualnych zastosowań.



Licznik zajmujący się czasem poświęconym na odpowiedź gracza, oraz warunek który wykonuje się wyświetlając komunikat koniec czasu, jeśli gracz nie wpisał żadnej odpowiedzi. Oraz warunek który sprawdza czy jest już koniec rozgrywki, czy 5 pytań już się wyświetliło. Jeśli jeden z klientów się opóźnia to poczeka na komunikat o werdykcie na drugiego, wyświetlając komunikat ”Oczekiwanie..”



Jeśli natomiast gracz odpowiedział to przesłana dowiedź jest sprawdzana, w zależności czy odpowiedź jest prawidłowa czy nie punkty są dodawane. Za sprawdzenie jest odpowiedzialna wcześniej wspomniana funkcja w klasie losowanie.



Wyświetlanie werdyktu oraz resetowanie zmiennych dzięki funkcjom , lub dodawanie do zmiennych odpowiednich wartości.