|  |  |  |
| --- | --- | --- |
| Aleksandra Duda | Gr. 31 lab 02 | Techniki Obiektowe |
| Nr indeksu: 130510 | Laboratorium nr 5 | 1.12.20r. |

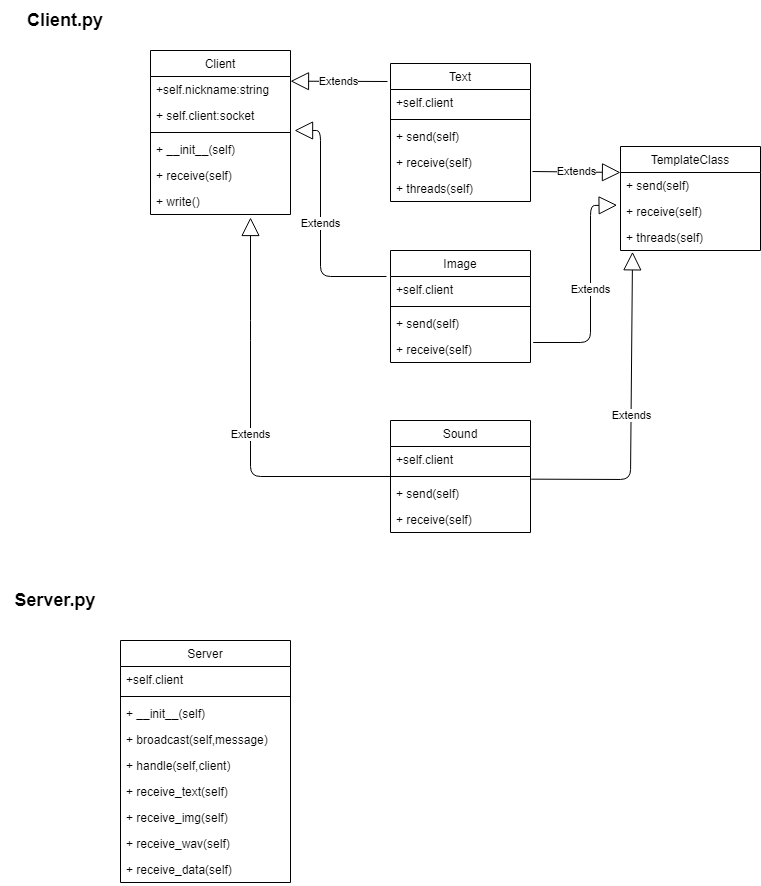
**Sprawozdanie z labolatorium nr 5**

1. **Cel ćwiczenia**

Transmisja danych.

Realizacja programu Client- Server w postaci czatu tekstowego z możliwością przesyłu plików dźwiękowych i obrazów.

1. **Model UML**



1. **Przebieg ćwiczenia**

**Server.py**

Najpierw zrobiłam implementację servera. Pełni on tutaj rolę naszego „Obserwatora” , nasłuchuje, czy jakiś klient się połączył i rozprowadza wiadomość do wszystkich podłączonych użytkowników.



W serverze zaimplementowałam metody takie jak **broadcast**  oraz **handle.** Metoda broadcast zapewnia to, że wiadomości są wysyłane do wszystkich Clientów, natomiast metoda **handle**  zapewniała między innymi informowanie o tym kiedy ktoś opuścił chat, czyli gdy jakiś Client się rozłączył, funkcjonuje jako taki **łańcuch zobowiązań.**



W tej klasie również zdefiniowałam metodę jak **receive\_data** , która rozpoznaje, którą opcję przesyłania wybrał Client i przekierowuje do odpowiednich funkcji. Serwer musi odebrać przesyłany plik od klienta. W przypadku , gdy jest to obraz to do **receive\_img**, gdy tekst to **receive\_text** , a gdy **receive\_wav** to jest odbierany dźwięk. Przy odbieraniu pliku najpierw jest tworzony nowy, po czym wpisywane są do niego dane. Stąd odebrany plik oznaczyłam jako **image\_received.**

**Client.py**



Klasa Client odpowiada za wysyłanie wiadomości i odbieranie jej od innych klientów. W metodzie statycznej **write**, a więc zanim zostanie stworzony obiekt najpierw rozpoznaję jaką wiadomość Client chce wysłać. Po czym przekierowuję do **metody szablonowej**.



Metoda **threads**  w klasie TemplateMethod to **metoda szablonowa.**

Z racji tego, że używałam wątków i w każdym z przypadków (tzn. Przesyłanie tekstu, obrazu, czy wav) czynności były powtarzalne. Najpierw któryś z Clientów wysyłał coś i wszyscy to odbierają, więc w tym przypadku najpierw ustaliłam, że pierwsze będzie odbieranie, żeby dane były jak najbardziej aktualne, a potem wysyłanie.

Natępnie zaimplementowałam klasy dla tekstu(klasa **Text**), obrazu(klasa **Image**) i dźwięku(klasa **Sound**).

Każda z nich dzedzicy po klasie **Client** i po klasie gdzie mamy **metodę szablonową.**************Implementacja dźwięku i obrazu już nieco się różni, bo należy wczytywać plik, korzystałam też z bibliotek i na pewno trzeba było sprawdzać rozmiar przesyłanego pliku, żeby wysłać odpowiednią liczbę

1. **Wnioski i działanie**

* W przypadku przesyłu plików,czy tekstu jest pewna powtarzalność , jest odbieranie wiadomosci i przesyłanie – mogłam tutaj wykorzystać metodę szablonową, dzięki której nie musiałam implementować tej metody w każdej klasie i tym samym kod jest bardziej przejrzysty.
* Kluczową rzeczą w tym projekcie było użycie gniazd – bez nich zreazlizowanie zadania dla mnie byłoby niemożliwe.
* Aby czat działał poprawnie należało zaimplementować w serverze metodę broadcast tak, aby każdemu użytkownikowi chatu wyświetlało się kto do niego dołączył i wszystkie wiadomości.
* Może być tylko jeden server, ale wielu klientów.
* Aby nie marnować pamięci RAM warto sprawdzać rozmiar ppliku i przesyłać dokładnie tyle ile trzeba (unikamy nadmiarowości )

Największy problem miałam problem ze zrozumieniem jak tutaj mają działać wątki. Było to dość trudne zagadnienie, bo wszystko dzieje się na bieżąco i musi ze sobą współpracować.

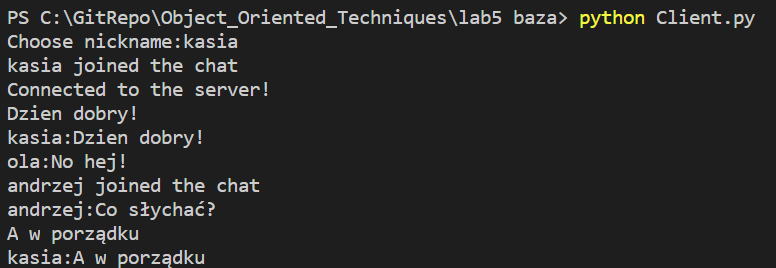
Na samym początku użytownik wybiera co chce zrobić: czy pisać, czy wysłać obraz lub plik dźwiękowy wav.

Aby nieco sobie ułatwić ustawiłam sobie po prostu jeden obraz i jeden plik dźwiękowy możliwy do przesyłania – sądzę, ze chodzi tu bardziej o istotę przesyłania plików w relacji Client-Server, aniżeli o to jaki konkretnie plik img czy wav zostanie przesłany.

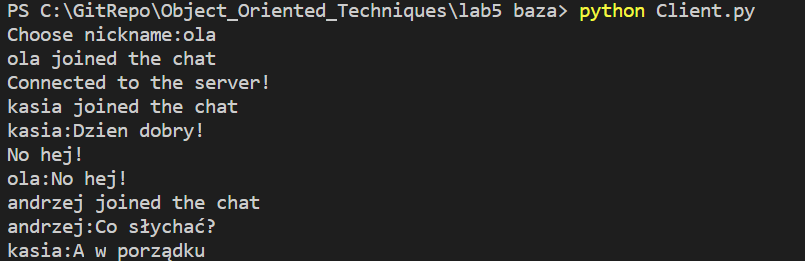


Użytkownik może wybrać opcję 1, 2 lub 3.

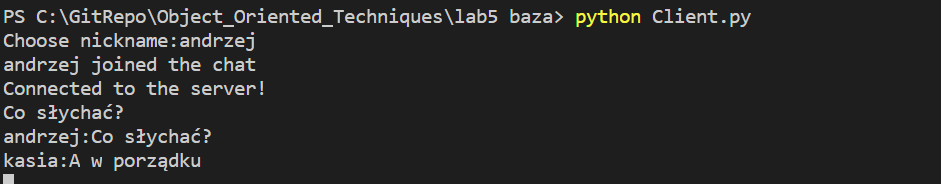
Na początku dołącza klient o nicku „kasia”



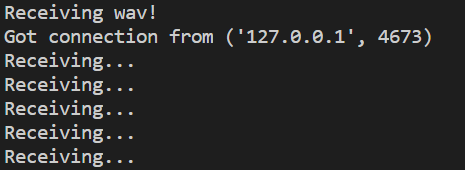
Następną osobą, która dołączyła do czatu była osoba o nicku „ola”



Natępnie na końcu „andrzej” dołączył do czatu:

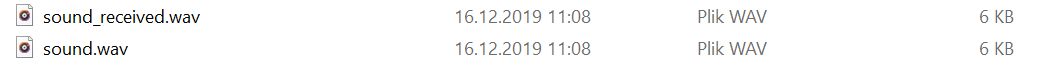


Jest także możliwość wysyłania plików:



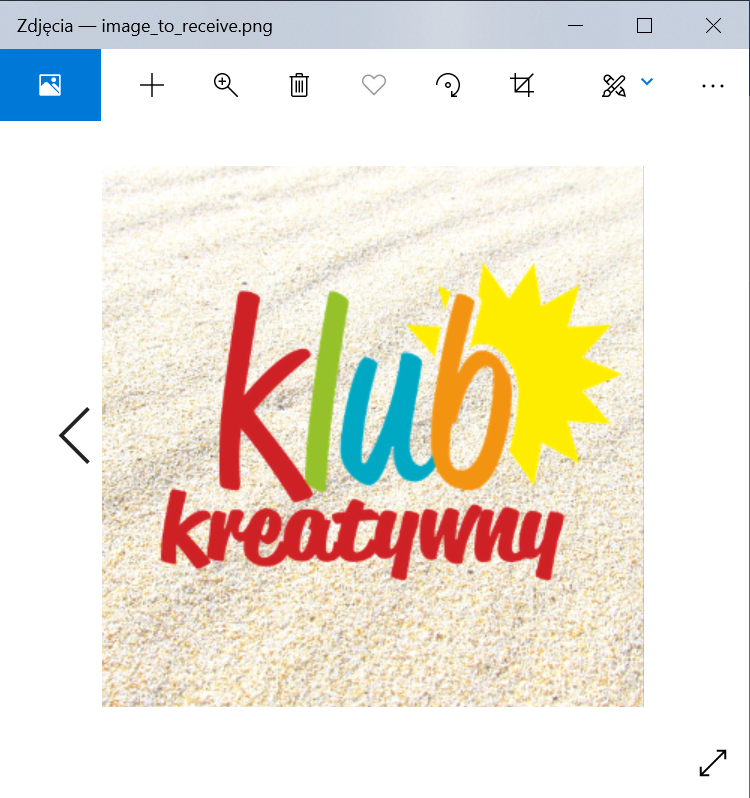
W tym przypadku wysyłam wav, jest zreazlizowane również wysyłanie obrazu z tym, że implementacja i rozmiar pliku się różnią, ale jakby samo przesyłanie obrazu w konsoli wygląda analogicznie do pliku wav.

Przesłane pliki mają rozmiary takie same, czyli wszystko się przesłało:





Otrzymany obraz **image\_to\_receive:**



Otrzymany dźwięk **sound\_received:**

