

Cient and Server

Projekt aplikacji

Wykonał:

Krzysztof Agaś-Lange

215668

1. Cel projektu

Celem projektu jest stworzenie aplikacji klient – server.

2. Wymagania co do aplikacji.

A. Należy opracować aplikacje kliencka i serwerowa przy podanych poniżej założeniach:


B. Aplikacja kliencka: - Uruchamiana jest z dwoma parametrami: nazwa użytkownika i ścieżka do lokalnego folderu - Każdy klient ma swój lokalny folder z plikami - Aplikacja obserwuje lokalny folder i reaguje na zmiany. Jak pojawia się tam nowe pliki, to wysyła je na serwer - Jak pojawi się nowy plik dla danego użytkownika, to jest on pobierany do lokalnego folderu - Aplikacja po uruchomieniu odpytuje serwer o nowe pliki i je ściąga - Wysyłanie / odbieranie dzieje się przy wykorzystaniu puli wątków - Aplikacja kliencka ma interfejs graficzny (np. Java FX) pokazujący w czasie rzeczywistym czym się w danej chwili zajmuje klient ("Pobieram...", "Wysyłam ...", "Sprawdzam") oraz wyświetlający listę aktualnych plików w lokalnym folderze. Panel graficzny ma umożliwić także udostępnienie danego pliku innemu użytkownikowi. Listę dostępnych użytkowników należy pobrać z serwera.

C. Serwer: - 5 folderów, które symulują 5 serwerów lub 5 dysków - Klient wysyła np. 10 plików, więc serwer uruchamia ileś wątków na których równolegle kopiuje pliki do tych dysków (folderów) - Wymagany jest kontroler, który tak rozłoży ruch, że do każdego z dysków (folderów) jednocześnie jest kopiowana taka sama liczba plików - Jeżeli podłączy się drugi klient, który zacznie wysyłać pliki, nie może on czekać aż skończą się zadania pierwszego klienta. Lista żądań na serwerze musi ulec reorganizacji, tak aby obydwaj klienci mieli wrażenie natychmiastowej obsługi (zaproponuj stosowny algorytm) - Na każdym dysku serwera znajduje się plik tekstowy (np. csv), w którym jest opisana zawartość danego dysku i kto jest jego właścicielem. Zauważ, że plik będzie uaktualniany przez wiele wątków. Rozwiąż ten problem. - W celu wizualizacji symulacji na niewielkiej liczbie użytkowników, czas kopiowania ma być sztucznie wydłużony poprzez usypianie wątku na losową liczbę sekund. - Serwer posiada panel graficzny (np. Java FX) pokazujący zawartość 5ciu dysków (serwerów) oraz aktualnie wykonywane operacje.

D. Zaproponuj listę testów jednostkowych opracowanych w JUnit, osobno dla klienta i serwera obejmujących kluczowe operacje wykonywane przez każdego z nich.

3. Server

Server jest włączany z dwoma parametrami do podania: **Portem serwera** oraz **Nazwą serwera**. Po wprowadzeniu danych i zatwierdzeniu ich przyciskiem **Start server** włączany jest server.

 Server

SERVER

Server's port:

Start server


Server's name:

Stop server

| File name | Owner | Shared | Users: |
|---------------------|-------|--------|--------|
| No content in table | | | |

| DYSK 1 | DYSK 2 | DYSK 3 | DYSK 4 | ALL |
|--------|--------|--------|--------|-----|
| - | - | - | - | - |
| 0 | 0 | 0 | 0 | 0 |

Pod główną tablicą można zobaczyć aktualnie zapisane pliki w serverze, ich właścicieli oraz do kogo są udostępnione. Po prawej stronie w tabeli **Users** można zobaczyć aktualnie połączonych użytkowników z serverem. Na dole znajduje się status 5 dysków oraz to, jaką ilość plików aktualnie posiadają i ile aktualnie jest do nich przesyłanych plików.

 Server

SERVER

Server's port:

Start server

Server's name:

Stop server

| File name | Owner | Shared | Users: |
|------------|-------|--------------------------|--------|
| Wpis.class | krizu | <input type="checkbox"/> | krizu |
| zad7.class | krizu | <input type="checkbox"/> | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| DYSK 1 | DYSK 2 | DYSK 3 | DYSK 4 | ALL |
|---------|---------|---------|---------|---------|
| 1 files | 1 files | 0 files | 0 files | 2 files |
| 0 | 0 | 0 | 0 | 0 |

4. Klient

Po uruchomieniu aplikacji klienta, aplikacja prosi o podanie **User name** oraz **Path to local folder** oraz **Server's port**, które są już domyślnie wypełnione, ale można je zmienić. Przycisk Connect powoduje uruchomienie aplikacji wraz z podanymi parametrami.

The screenshot shows the Klient application window. It has a header with a user icon and the word 'KLIENT'. Below the header, there are three input fields: 'User name:' (empty), 'Path to local folder:' (containing 'folder_krizu'), and 'Server's port:' (containing '55555'). To the right of these fields are four buttons: 'Share' (disabled), 'Connect' (green), 'Unshare' (disabled), and 'Disconnect' (red). Below the buttons is a table with four columns: 'Name', 'Owner', 'Shared to', and 'Status'. The table is empty, and a message 'No content in table' is displayed in the center. At the bottom of the window, there is a 'Clear console' button.

Pod główną tablicą można zobaczyć pliki u danego użytkownika, kto jest ich właścicielem, do kogo są udostępnione oraz ich status wysłania. Na dole znajduje się konsola, na której pokazywane są najważniejsze informacje związane z danymi akcjami. Za pomocą przycisków **Share** i **Unshare** można udostępnić plik innemu użytkownikowi jak i to cofnąć. **Przycisk Clear** console służy do czyszczenia konsoli.

The screenshot shows the Klient application window after connecting to a server. The 'User name' field now contains 'krizu'. The 'Share' button is now enabled. The table below the buttons contains three rows of data:

| Name | Owner | Shared to | Status |
|-----------------------|-------|-----------|----------------|
| zad7.class | krizu | [krizu] | local + server |
| Wpis.class | krizu | [] | local + server |
| Clienttxt — skrót.Ink | krizu | [] | local |

Below the table is a console window showing the following output:

```
share : ('Clienttxt — skrót.Ink',0,'folder_krizu\\krizu\\Clienttxt — skrót.Ink','krizu',[],1178)
~: Clienttxt — skrót.Ink (1178B) ...
```

At the bottom of the window, there is a 'Clear console' button.