ServerCommunicator

Instrukcja obsługi biblioteki

Po pierwsze, aby w ogóle nawiązać połączenie, należy zdobyć instancję klasy Client.

import client.communication.Client;

(…)

Client c = Client.getInstance();

Sprawdź co otrzymałeś! Metoda może zwrócić null’a!!! Dzieje się tak, kiedy połączenie nie może zostać nawiązane! Jeżeli jednak otrzymałeś obiekt klasy klient, ciesz się, gdyż połączenie już zostało nawiązane! ☺

Teraz za pewne chciałbyś się zalogować. Metoda logIn() przyjmuje 3 argumenty: login, hasło i obiekt ThreadLoginEvent. Aby takowy obiekt zdobyć potrzebujesz stworzyć własną klasę i dziedziczyć w niej klasę client.event.TheadLoginEvent . W klasie tej zaś powinieneś nadpisać metodę proces(). Przykład tego nadpisania znajduje się poniżej:

@Override

**public** **void** process()

{

**if** (**this**.success)

System.*out*.println("Successfully logged in");

**else**

System.*out*.println("Failed to log in");

}

Jako, że nie możemy blokować wątku, w którym działa interfejs, metoda logIn() uruchamia osobny wątek, który komunikuje się z serwerem i siłą rzeczy nie może sama zwracać wartości. Dlatego też zaraz po otrzymaniu odpowiedzi od serwera uruchamiana jest metoda proces z ThreadLoginEvent.

W przypadku każdej klasy implementującej interfejs ThreadEventProcessor konieczne jest też zaimplementowanie metody errorOccured(). Jest ona wołana w przypadku wystąpienia błędu podczas nawiązywania komunikacji z serwerem.

Poniżej znajduje się pełen przykład wołania metody requestUnassignedParcels(), która również wymaga podania klasy ThreadListResponseEvent, która tak samo jak ThreadLoginEvent implementuje interfejs ThreadEventProcessor.

**public** **static** **void** main(String [] args)

{

Client c = Client.*getInstance*();

c.requestUnassignedParcels(**new** ThreadListResponseEvent(){

@Override

**public** **void** process() {

**for** (String [] s : list)

System.*out*.println(s[0]);

}

@Override

**public** **void** errorOccured() {

System.*out*.println("Error");

}});

}