Programare avansata pe obiecte

Lab 2-3-4

(Paduraru Ciprian)

1. Explicatie Sockets la nivel de SO. Diferente TCP/UDP.

2. Implementare in Java. Urmariti si executati programele din Socket1. Rulati initial serverul, apoi mai multe aplicatii client.

3. **Aplicatie 1**: Implementati un server care sa efectueze taskuri matematice simple:

- ridicare la putere

- suma a N numere.

Clientii pot face requesturi de mai multe tipuri de operatii folosind linia de comanda (deocamdata considerati un singur client capabil sa se conecteze la un moment dat la server).

* Deconectare (“disconnect”).
* Una din:
  + add N a1 a2 ....aN
  + mul N a1 a2....aN
  + pow A B

Serverul trimite mesaje inapoi clientului cu rezultatul.

(O posibila solutie este implementata in proiectul Sockets2).

**Hints**:

1. Serializare

Pentru a putea trimite eficient mesaje cu o semantica in functie de tipul mesajului avem nevoie de serializare. Comunicarea intre client-server se va face prin bytes streamuri. Java suporta serializarea prin mostenirea din “Serializable”. Toate obiectele agregate in clasa derivata trebuie sa poata fi serializabile, creandu-se astfel un graf de serializare.

O posibila imp pentru clasa mesaj (client -> server) si raspuns (server -> client):

public class **Message** implements Serializable

{

public enum MsgType { MSG\_INVALID, MSG\_DISCONNECT, MSG\_ADD, MSG\_MUL, MSG\_POW };

public MsgType mType;

Message() { mType = MsgType.MSG\_INVALID; mN = 0; }

// TODO: as you can see ideally we should have a base class for message then multiple derived class depending on message type

public int mN;

public int[] mNumbers;

public int mA, mB;

}

public class TaskResult implements Serializable

{

public int mResult;

}

1. Trimiterea / primirea mesajelor

Dupa creerea socketului (cod server sau client) putem obtine stream-ul de input/output al socketului folosind:

ObjectOutputStream streamToServer = new ObjectOutputStream(**clientSocket**.getOutputStream());

ObjectInputStream streamFromServer = new ObjectInputStream(**clientSocket**.getInputStream());

Pentru a scrie / citi un obiect de pe stream: streamToServer.**writeObject**(msg); / (Message) streamToServer.**readObject**();

1. Citirea informatiilor de la consola.

Pe client, putem citi comenzile date linie cu linie si pentru fiecare linie citita sa putem returna o instanta de Message:

Scanner scan = new Scanner(System.in);

String msgText = scan.nextLine();

final String[] tokens = msgText.split(" ");

// Process tokens to get array

4. Sockets non-blocanti vs threads. Ce s-ar intampla in aplicatia de mai sus daca am dori sa procesam taskuri de la N clienti si de asemenea sa avem o consola de comenzi si pentru server ?

* Modificati aplicatia precedenta avand o clasa ClientHandler derivata din Threads. In ea, pastrati socketul unui client si adresa IP al acestuia ca string.
* Pe server, pastrati un ArrayList < ClientHandler>. Aveti grija la operatiile de add si remove din aceasta lista care trebuie sa se faca sincronizat ( thread-ul de Server poate rula in paralel cu thread-ul ClientHandler... ).
* Implementati o clasa ServerConsole derivata din Threads care sa ruleze pe Server in paralel cu operatiile de aceptare a clientilor (listen ). Singura comanda acceptata este “showClients” ce arata numarul de conecti conectati la server si IP-urile acestora.
* Testati aplicatia cu mai multi clienti.

(Puteti consulta solutia din **Socket3** pentru rezolvare)

5. Aplicatie:

- Urmariti codul pentru javaTicTacToe si intelegeti cum functioneaza.

- Completati codul pentru a putea juca intre 2 jucatori online. Considerati 2 arhitecturi:

* Unul dintre playeri devine server, iar celalalt se conecteaza la adresa lui IP.
* Server dedicat. Toate mesajele de pe cei 2 clienti ajung la server si acesta modifica status-ul.

Nice to have:

- Ping system using UDP pentru a detecta timeout-ul.

- Folositi broadcast pentru a putea identifica procesele din LAN care ruleaza aceeasi aplicatie.

7. Explicatii UDP/TCP/multiplayer games programming

- Strategy games, FIFA syncronization: Lockstep with buffering

- General shooter games

- Racing games syncronization.

8. Discutie proiecte / hinturi.