Ejemplo de como podría ser el examen de Dad

Probablemente se nos pida algo del estilo a la implementación de un protocolo, en el que un servidor deberá recibir comandos de un cliente y responder en función a estos.

Elemento

Este será la clase con el que trabajaremos:

```
public class Elemento{
  //Atributos:
  String nombre
  double cantidad;
  double veces = 0;

  //Constructor:
  public Elemento(String nombre, double cantidad){
    this.nombre = nombre;
    this.cantidad = cantidad;
  }
}
```

Cliente

Este será el cliente que utilizaremos para enviar señales al serverSocket, este cliente permanecerá activo permitiendo enviar mensajes al servidor hasta que se envíe el comando EXIT y se reciba la señal “cerrado” del servidor.

```
public class Cliente{
  try{//Se debe hacer siempre try Catch cuando andamos con sockets
    socket = new Socket("localhost", 5000);//Inicializamos nuevo socket con
IP y Puerto
    //Buffers de lectura y escritura:
    BufferedReader br = new BufferedReader(new
InputStramReader(socket.getInputStream()));
    PrintWriter pw = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));
    Scanner sc = new Scanner(System.in);//Para leer por teclado
    String rl = ""; //Aqui guardamos cada línea leída
    do{//repetimos contenido hasta que se reciba "Cerrado"
      pw.println(sc.nextLine());//Leemos del teclado con sc y enviamos al
servidor con pw
      pw.flush(); Limpiamos salida
```

```
      //Recibimos línea del servidor con br, la almacenamos en rl y la
mostramos en pantalla:
      System.out.println((rl = br.readLine()))
    }while(lineaLeida.contentEquals("Cerrado")==false)//Mientras no se
reciba una señal de cierre
  }catch(IOException e){//en caso de salir excepción
    e.printStackTrace();
  }
}

public static void main(String[] args){
  (new Cliente()).ejecutar();
}
```

Servidor

```
public class Server{
  ServerSocket serverSocket;//Declaramos Socket Servidor
  Socket socket;//declaramos Socket standar
  Static Hashtable<string, ArrayList<elemento>> listaElementos = new
Hastable<String, ArrayList<elemento>>;//Lista de elementos
  public void ejecutar(){
    try{
      serverSocket = new ServerSocket(5000) //Indicamos puerto de escucha
      while(true){
        socket=serverSocket.accept();//Iniciamos escucha
        (new ServerThread(socket)).start()//Arrancamos el servidor en un
hilo;
      }
    }catch(IOException e){
      e.printStackTrace()//mostramos excepción ocurrida
    }
  }
  public static void main(String[] args){
    (new Server()).ejecutar(); //Iniciamos el hilo del servidor
  }
}
```

ServerThread

En el ServerThread extendemos la funcionalidad de Thread e implementamos la funcionalidad del server Socket. En este caso lo que hará nuestro servidor es reaccionar a los comandos que se le manden y en caso de recibir un comando inválido enviar un mensaje indicando que el comando no es válido.

From:
https://www.knoppia.net/ - **Knoppia**

Permanent link:
https://www.knoppia.net/doku.php?id=dad:ejercicioprotocolos&rev=1699870559

Last update: **2023/11/13 10:15**