

# Software Distribuït

**SESSIÓ 3: Pràctica 1 - Client i Servidor**

# Proposta de control paràmetres d'entrada

```
HashMap<String,String> options = new HashMap();

for (int i=0; i<args.length; i=i+2)
{
    options.put(args[i],args[i+1]);
}

try{
    hostname = options.get("-s");
    port = Integer.parseInt(options.get("-p"));

    if (options.containsKey("-i")
    {
        //...//
    }

} catch{
    //...//
}
```

# Estructuració del codi

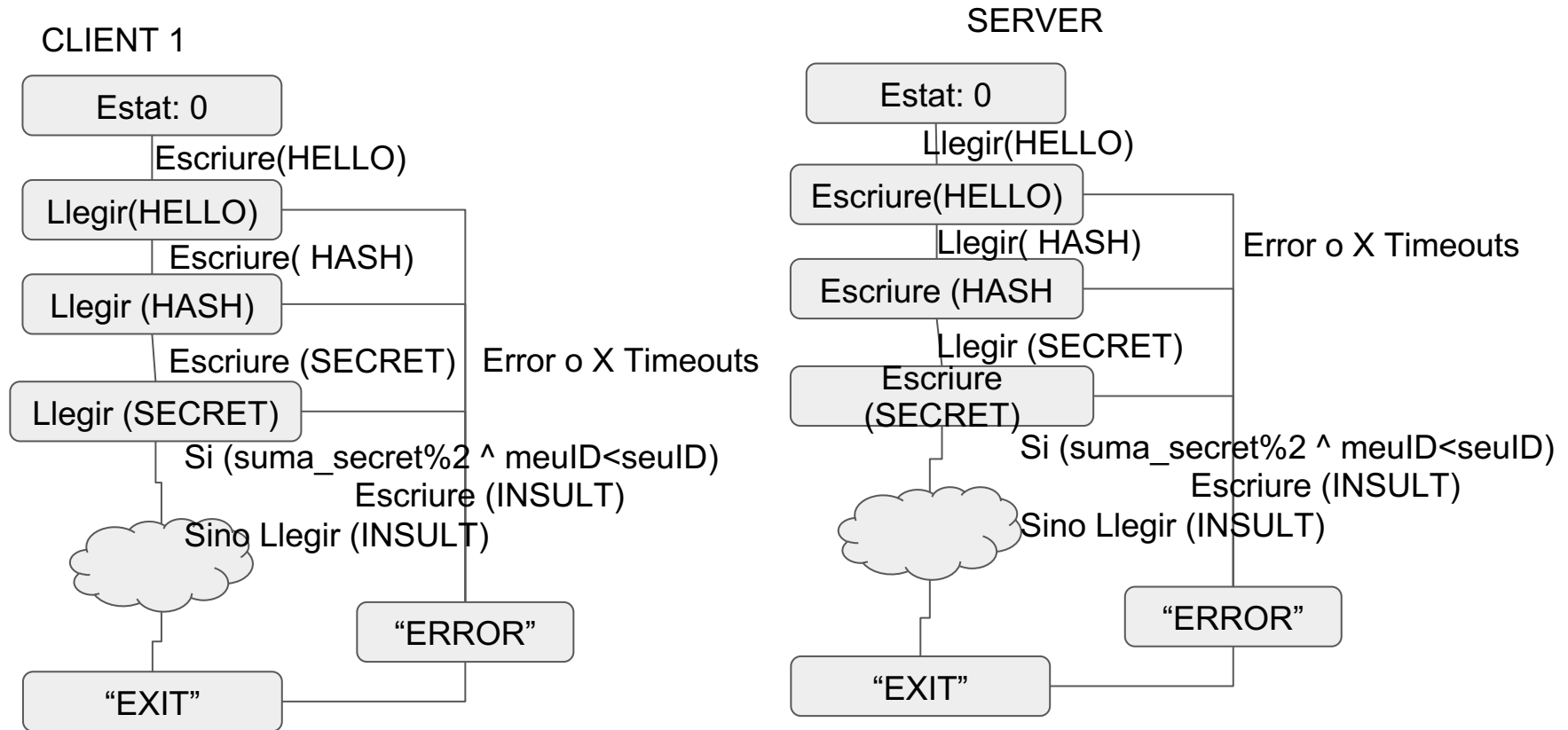
Encapsulament i estructuració les funcions a realitzar:

- Ampliar ComUtils amb les funcions de baix nivell adequades al protocol (read\_opcode, read\_char, read\_hash, write\_...).
- Funcions de més alt nivell lligades al protocol (insultar, contar punts...)
- Classes i funcions lligades al funcionament del joc (màquina d'estats)
- Funcions de la lògica (get\_winner, IA, etc.)

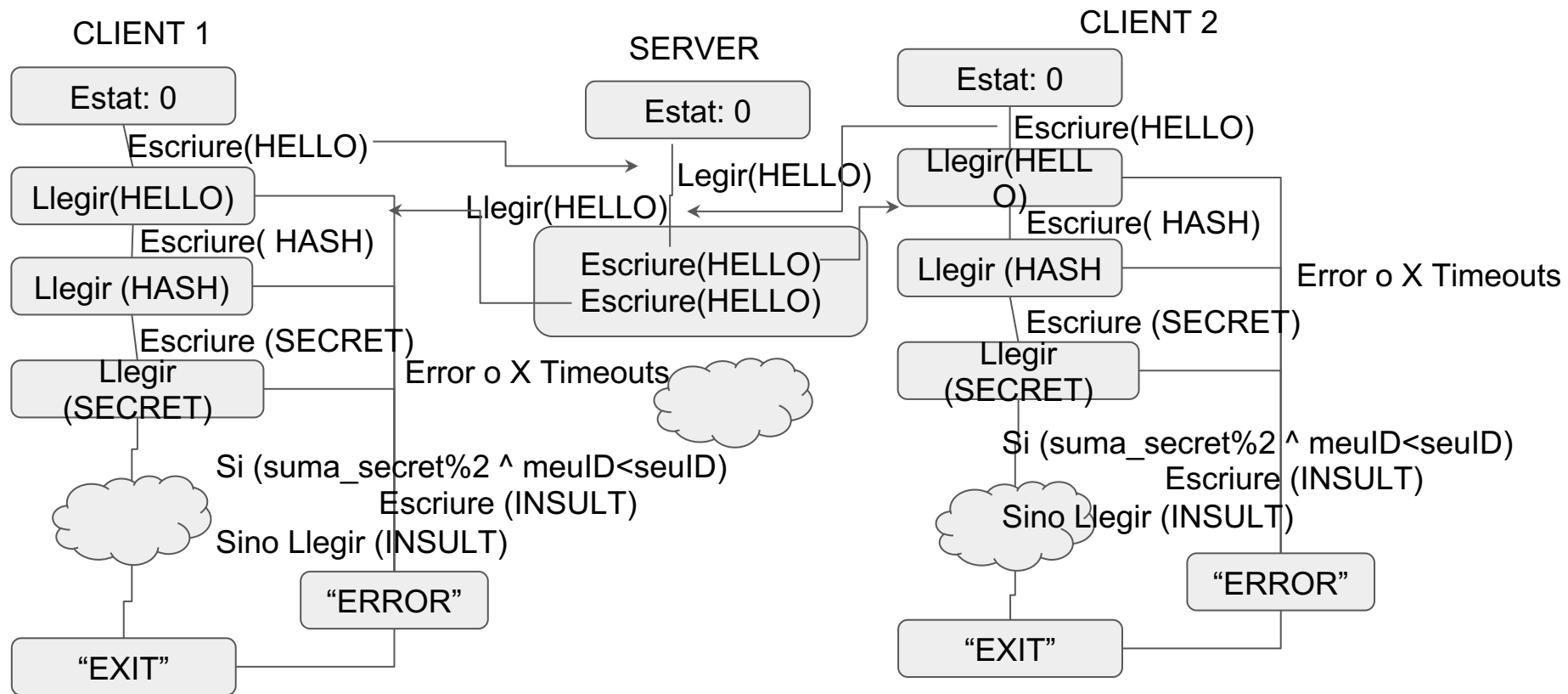
**Intentar separar al màxim el protocol de la lògica del joc.**

**Comproveu els tipus de DADES de les TRAMES que s'enviaran per SOCKET amb el PRTOCOL!!!**

# Disseny de la lògica (màquina d'estats) 1player



# Disseny de la lògica (màquina d'estats) 2 Player



# Exemple Client

```
public class Client {  
  
    public static void main(String[] args){  
  
        //Tractament de paràmetres de consola  
  
        try{  
            Socket socket = new Socket(nomMaquina, numPort);  
            socket.setSoTimeout(500); //en ms.  
  
        } catch (IOException e) {  
            System.out.println("IOException: "+ e.getMessage());  
        }  
  
        //...//  
    }  
}
```

# Exemple Server

```
public class Server{

    public static void main(String[] args) throws IOException {
        //Tractament de paràmetres de consola

        try{
            ServerSocket serverSocket = new ServerSocket(numPort);
            Socket s = serverSocket.accept();
            s.setSoTimeout(500);

            //...//

        }catch (IOException e) {
            System.out.println("IOException: "+ e.getMessage());
        }

    }
}
```

# Exemple Server Multithread

```
public class ServerMT{
    public static void main(String[] args) throws IOException {
        //Tractament de paràmetres de consola
        try{
            ServerSocket serverSocket = new ServerSocket(numPort);

            while(true){
                Socket s = serverSocket.accept();
                s.setSoTimeout(500) //en ms.
                new Thread(new Game(s, var1, var2)).start()
            }

        }catch (IOException e) {
            System.out.println("IOException: "+ e.getMessage());
        }
    }
}
```



# ComUtils amb sockets

```
public ComUtils(InputStream inputStream, OutputStream outputStream) throws
IOException {
    dataInputStream = new DataInputStream(inputStream);
    dataOutputStream = new DataOutputStream(outputStream());
}

public ComUtils(Socket socket) throws IOException {

    /.../

    dataInputStream = new DataInputStream(socket.getInputStream());
    dataOutputStream = new DataOutputStream(socket.getOutputStream());

    /.../
}
```

# Exemple Server Multithread de 2 jugadors

```
public class ServerMT2P{
    public static void main(String[] args) throws IOException {
        //Tractament de paràmetres de consola
        try{
            ServerSocket serverSocket = new ServerSocket(numPort);

            while(true){
                /*Esperar a dos sockets disponibles*
                Socket s1 = serverSocket.accept();
                Socket s2 = serverSocket.accept();
                new Thread(new Game(s1,s2, var1, var2).start())
            }

        }catch (IOException e) {
            System.out.println("IOException: "+ e.getMessage());
        }
    }
}
```

# Threads

Hi ha dues maneres d'utilitzar Threads:

```
public class ServerThread implements Runnable {  
    public void run() {  
        //...//  
    }  
}  
  
public class ServerThread extends Thread {  
    public void run() {  
        //...//  
    }  
}
```

# Threads

```
public class ServerThread implements Runnable
```

```
Thread server = new Thread(new ServerThread());  
server.start();
```

```
public class ServerThread extends Thread
```

```
Thread server = new ServerThread();  
server.start();
```

# Exceptions Custom

**Declarar la classe error com a extends Exception :**

```
public class MalformedString extends Exception {  
  
    public MalformedString (String message) {  
        super(message);  
    }  
}
```

# Exceptions Custom

## 2. Utilitzar-la en alguna funció/mètode que “llenci” l'excepció :

```
public class Server {  
    public read_show() throws MalformedString{  
        if(!check_command("SHOW"))  
        {  
            throw new MalformedString("Command error: " +  
"SHOW" );  
        }  
    }  
}
```

# Exceptions Custom

## 3. Try / Catch

```
try{  
    server.read_show()  
}  
  
catch (MalformedString ex)  
{  
  
}
```

# Execució del codi

Característiques definides per l'execució del codi:

servidor> java -jar server.jar -h **(ha de mostrar un help)**

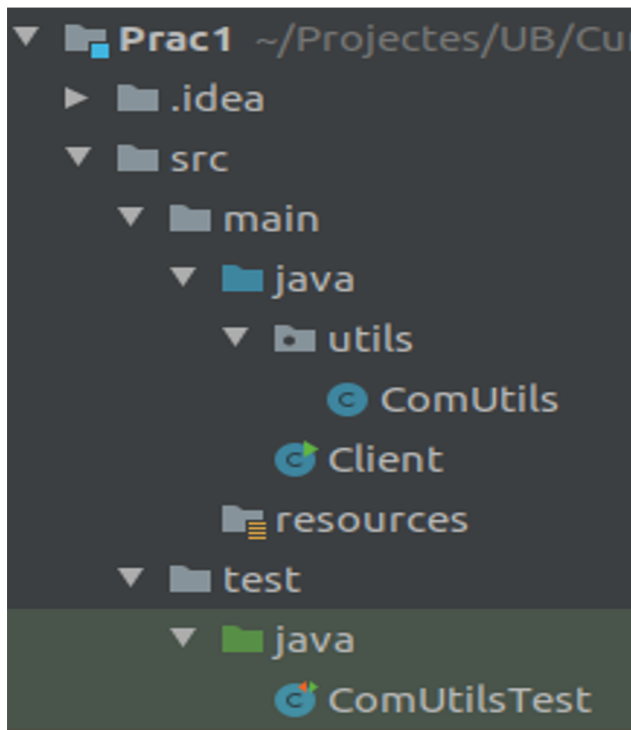
Us: java -jar server.jar -p <port> [-m 1|2] **(ha de seguir aquest format en aquest ordre i detectar errors)**

client> java -jar client.jar -h **(ha de mostrar un help)**

Us: java -jar client.jar -s <maquina\_servidora> -p <port> [-i 0|1] **(ha de seguir aquest format en aquest ordre i detectar errors)**



# Estructura de projecte



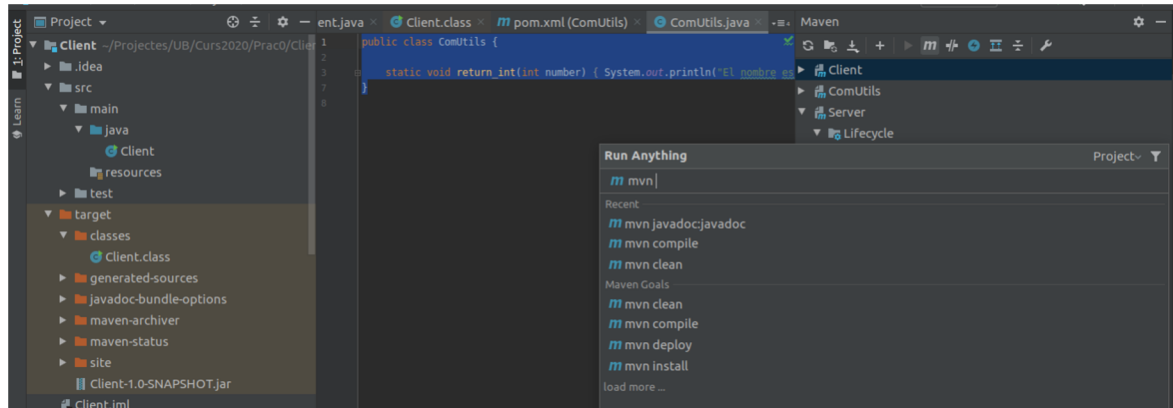
Es proposa una estructura senzilla on es separa en dos projectes Client i Server.

He tenir en compte:

- S'ha de poder executar per terminal a través de mvn i desitjablement amb java.
- Si es segueix una altra estructura especificar la compilació i execució del projecte en la seva entrega (README).

# Funcionalitats Maven

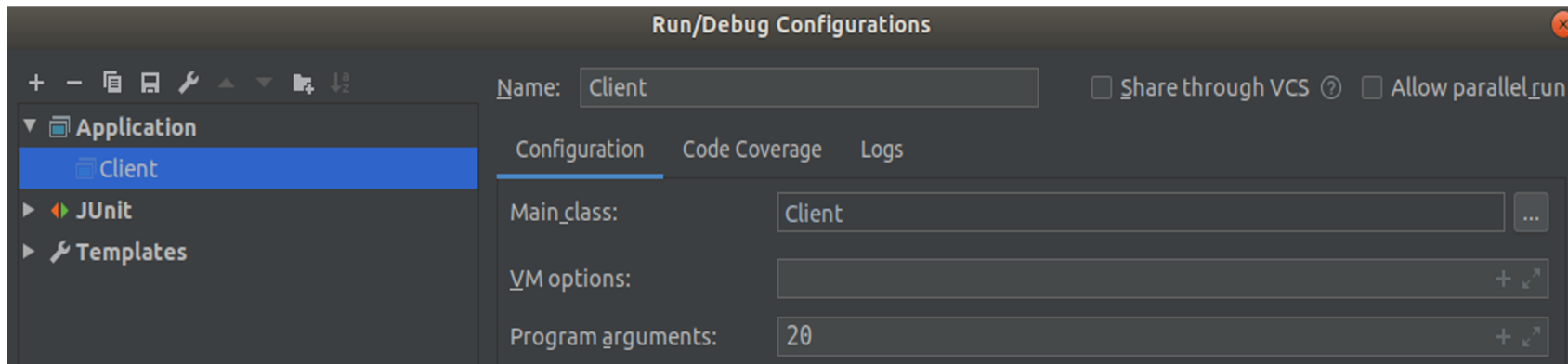
- **mvn clean** : configuració del projecte en un equip nou o per modificació del pom.xml
- **mvn compile** : compilació del codi/generació dels .class
- **mvn test** : execució dels tests
- **mvn javadoc:javadoc** : generació de javadoc



View -> Tool Windows -> Maven

# Executar amb parametres

- **IntelliJ:** Run -> Edit Configurations



Després “Run Client”

- **Maven:** `mvn mvn exec:java -Dexec.mainClass=Client -Dexec.args="20"`
- **Java:** `cd /target/classes`  
`Java Client 20`