Activitat 3

* **Autoavaluació:**

Aquesta activitat creiem que ens ha sortit bastant bé ja que teníem l’activitat passada feta i ens ha sigut més ja que teníem una bona base.

1. 2/2
2. 2/2
3. 2/2
4. 2/2
5. 2/2

Tot i tindre alguna que altre complicació creiem que ens ha sortit força bé

* **Codi de l’aplicació:**

1. **Classe, Act4\_Client:**

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.Random;

public class Act4\_Client {

public static int heartbeat;

public static String nom;

@SuppressWarnings("InfiniteLoopStatement")

public static void main(String[] args) {

heartbeat = 2 + (new Random().nextInt(5) + 1);

System.out.println("Aquest client ha decidit fer un heartbeat de " + heartbeat + " segons");

nom = "client\_" + generate();

Act4\_Server\_Interface serv = null;

try {

Registry registry = LocateRegistry.getRegistry("localhost", 5555);

serv = (Act4\_Server\_Interface) registry.lookup("Act4");

} catch (Exception e) {

e.printStackTrace();

}

if (serv != null) {

try {

serv.anunciarPresencia(nom, heartbeat);

} catch (RemoteException e) {

e.printStackTrace();

}

}

for (;;) {

long startTime = System.nanoTime();

long targetTime = startTime + (heartbeat \* 1\_000\_000\_000L);

for (;;) {

if (System.nanoTime() >= targetTime) {

System.out.println("Batec!");

try {

assert serv != null;

serv.batec(nom);

} catch (RemoteException e) {

throw new RuntimeException(e);

}

break;

}

}

}

}

public static String generate() {

String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

Random rand = new Random();

StringBuilder result = new StringBuilder();

for (int i = 0; i < 5; i++)

result.append(alphabet.charAt(rand.nextInt(alphabet.length())));

return result.toString();

}

}

1. **Classe, Act4\_Server\_Interface:**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface Act4\_Server\_Interface extends Remote {

void anunciarPresencia(String nomClient, int heartbeat) throws RemoteException;

void batec(String nomClient) throws RemoteException;

}

1. **Classe, Act4\_Server implements:**

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

public class Act4\_Server implements Act4\_Server\_Interface {

public static List<Client> taula;

@Override

public void anunciarPresencia(String nomClient, int heartbeat) throws RemoteException {

taula.add(new Client(nomClient, heartbeat));

System.out.println(

"Nou client afegit a la taula; es diu " + nomClient + " i te un heartbeat de " + heartbeat + "s");

}

@SuppressWarnings({ "StringOperationCanBeSimplified", "ForLoopReplaceableByForEach" })

@Override

public void batec(String nomClient) throws RemoteException {

for (int i = 0; i < taula.size(); i++)

if (taula.get(i).nom.toString().equals(nomClient)) {

taula.get(i).estat = true;

taula.get(i).passadesServidor = 0;

System.out.println("Batec rebut desde " + nomClient);

}

}

@SuppressWarnings({ "RedundantCast", "InfiniteLoopStatement" })

public static void main(String[] args) {

Registry reg = null;

taula = new ArrayList<>();

try {

reg = LocateRegistry.createRegistry(5555);

} catch (Exception e) {

System.out.println("ERROR: Registry cannot be created.");

e.printStackTrace();

}

Act4\_Server serverObject = new Act4\_Server();

// Cast serverObject with Remote

try {

assert reg != null;

reg.rebind("Act4", (Act4\_Server\_Interface) UnicastRemoteObject.exportObject(serverObject, 0));

} catch (Exception e) {

System.out.println("ERROR: Server object cannot be registered.");

e.printStackTrace();

}

for (;;) {

long startTime = System.nanoTime();

long targetTime = startTime + 1\_000\_000\_000L;

for (;;)

if (System.nanoTime() >= targetTime) {

actualitzarTaula();

break;

}

}

}

@SuppressWarnings("ForLoopReplaceableByForEach")

public static synchronized void actualitzarTaula() {

for (int i = 0; i < taula.size(); i++)

if (taula.get(i).estat) {

taula.get(i).passadesServidor++;

if (taula.get(i).passadesServidor >= (taula.get(i).heartbeat \* 2)) {

taula.get(i).passadesServidor = (taula.get(i).heartbeat \* 2);

taula.get(i).estat = false;

System.out.println("Fa " + (taula.get(i).heartbeat \* 2) + "s que no rebo res desde "

+ taula.get(i).nom + ". El marco com inactiu.");

}

}

}

}

1. **Classe, Client:**

public class Client {

public String nom;

public int heartbeat;

public int passadesServidor;

public boolean estat;

public Client() {

}

public Client(String nom, int heartbeat) {

this.nom = nom;

this.heartbeat = heartbeat;

passadesServidor = 0;

estat = true;

}

public Client(String nom, int heartbeat, int passadesServidor, boolean estat) {

this.nom = nom;

this.heartbeat = heartbeat;

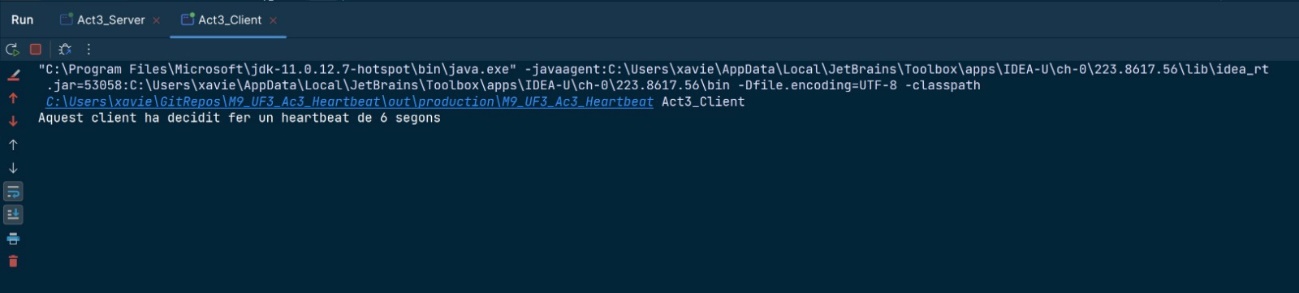
this.passadesServidor = passadesServidor;

this.estat = estat;

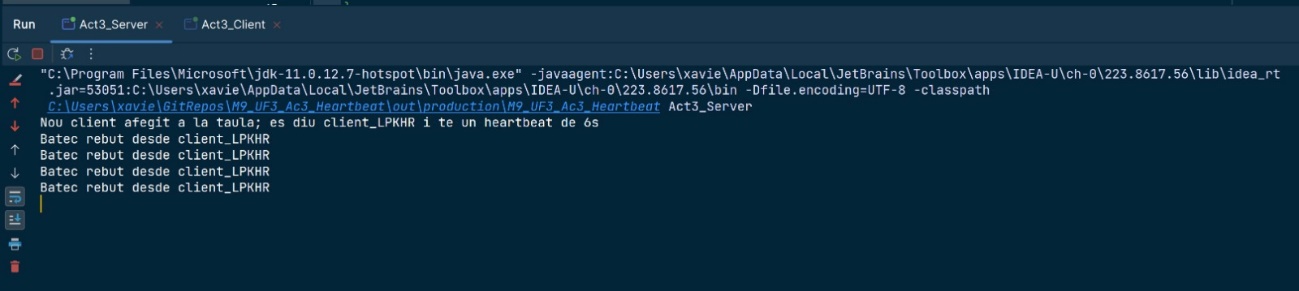
}

}

* **Captures de pantalla**
* Aquí tenim una captura del client iniciat



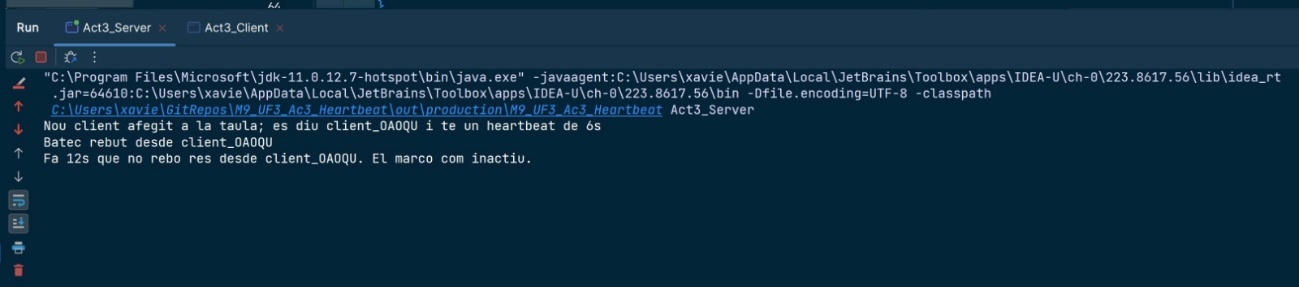
* Aquí tenim una captura del servidor iniciat



* En aquesta captura podem veure el server desde molts clients nous oberts



* En aquesta captura podem veure el server amb un client que passa a ser inactiu



* En aquesta captura podem veure el server amb un sol client que es desactiva i després es reactiva i torna a la vida

