Actividad para PSP04

Ejercicio 1

```
public class Servidor {
    private ServerSocket serverSocket;
    private final static int PUERTO = 2000;
    private final static int MAX_CLIENTES = 10;

    public Servidor() {
        ExecutorService executor = Executors.newFixedThreadPool(nThreads:MAX_CLIENTES);
        try {
            serverSocket = new ServerSocket(port:PUERTO);
            System.out.println("Servidor escuchando por el puerto: " + PUERTO);
            int clientesAtendidos=0;
            while(clientesAtendidos<MAX_CLIENTES){
                Socket clienteSocket = serverSocket.accept();
                System.out.println("Nuevo cliente aceptado:" + clienteSocket);
                executor.execute(new clienteHandler(clienteSocket));
                clientesAtendidos++;
                if(clienteHandler.finjuego()){
                    clienteSocket.close();
                    serverSocket.close();
                }
            }

        } catch (IOException ex) {
            ex.printStackTrace();
        }finally{
            executor.shutdown();
        }
    }
```

Para este ejercicio hicimos uso de la clase Executor Service para limitar el numero de conexiones de los clientes a 10

```
    private static class clienteHandler implements Runnable {



        private Socket clienteSocket;
        private int numeroSecreto;
        private static boolean fin=false;

        public clienteHandler(Socket clienteSocket) {
            this.clienteSocket= clienteSocket;
            numeroSecreto = (int) (Math.random()*101);
        }

        @Override
        public void run() {
            System.out.println("Cliente " + clienteSocket + " se ha conectado");
            try {
                DataInputStream in = new DataInputStream(in: clienteSocket.getInputStream());
                DataOutputStream out = new DataOutputStream(out:clienteSocket.getOutputStream());
                String numero;
                out.writeUTF("Te has conectado ocn el servidor correctamente \n"
                        + "escribe un numero");
                while((numero = in.readUTF())!= null){
                    int n = Integer.parseInt(s: numero);
                    if(n == numeroSecreto){
                        fin = true;
                        out.writeUTF("Felicidades has acertado el numero \n"
                                + "Se ha terminado el juego, te has desconectado");
                        break;
                    }else if(n < numeroSecreto){
                        out.writeUTF(str:"el numero es mayor");
                    }else{
                        out.writeUTF(str:"El numero es menor");
                    }
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }

        }
```

Creamos una clase que extienda de Runnable para que genere un hilo por cada cliente que se conecte al servidor

Ejercicio 2

```java
public class Server {

    private ServerSocket serverSocket;
    private final static int PUERTO = 1400;
    private DataInputStream in;
    private DataOutputStream out;

    public Server() {
        ExecutorService ex = Executors.newFixedThreadPool(nThreads: 10);
        try {
            serverSocket = new ServerSocket(port: PUERTO);
            System.out.println("Escuchando por le puerto " + PUERTO);
            int cli = 0;
            while (cli < 10) {
                Socket cliente = serverSocket.accept();
                System.out.println(x: "Nuevo cliente conectado");
                ex.execute(new clienteHandler(clienteS: cliente));
                cli++;
            }

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            ex.shutdown();
        }

    }
```

Establecemos un limite de conexiones de clientes a nuestro servidor

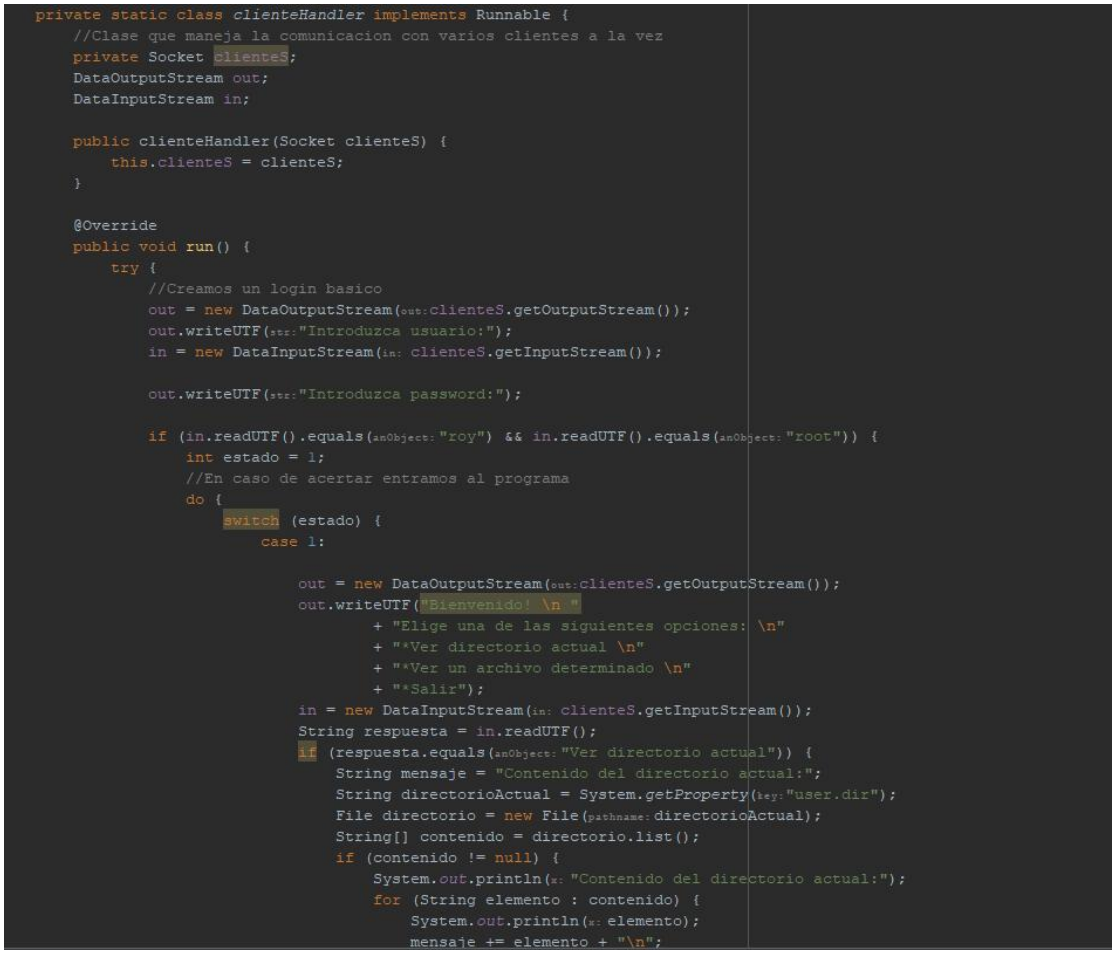
```java
    private static class clienteHandler implements Runnable {

        private Socket clienteS;

        public clienteHandler(Socket clienteS) {
            this.clienteS = clienteS;
        }

        @Override
        public void run() {
            System.out.println("Cliente " + clienteS + " se ha unido");

            try {
                DataInputStream in = new DataInputStream(in: clienteS.getInputStream());
                DataOutputStream out = new DataOutputStream(out:clienteS.getOutputStream());
                out.writeUTF("Te has conectado al servidor correctamente \n"
                        + "Introduce la ruta que quieras visualizar");
                String ruta = in.readUTF();
                BufferedReader br = new BufferedReader(new FileReader(fileName: ruta));
                String lectura;
                String archivo = "";
                while ((lectura = br.readLine()) != null) {
                    archivo += lectura + "\n";
                }
                out.writeUTF(str:archivo);
            } catch (IOException ex) {
                ex.printStackTrace();
            }

        }
    }
```

Y dejamos que los hilos gestionen las peticiones de los clientes en este caso se generara un hilo para que busque el fichero que el cliente solicita

Ejercicio 3

```java
public class Servidor {

    //Creamos el socket para el servidor
    private ServerSocket serverSocket;
    //El puerto por el que estara escuchando
    private final static int PUERTO = 1400;
    //Clases para comunicarnos entre el cliente y servidor
    private DataInputStream in;
    private DataOutputStream out;

    public Servidor() {
        //Clase que ejecuta hilos
        ExecutorService ex = Executors.newFixedThreadPool(nThreads: 10);
        try {
            //Iniciamos el servidor
            serverSocket = new ServerSocket(port: PUERTO);
            System.out.println("Escuchando por le puerto " + PUERTO);
            int cli = 0;
            while (cli < 10) {
                Socket cliente = serverSocket.accept();
                System.out.println(x: "Nuevo cliente conectado");
                ex.execute(new clienteHandler(clienteS: cliente));
                cli++;
            }

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            ex.shutdown();
        }

    }
```

Realizamos la misma configuración que en los ejercicios anteriores



```
private static class clienteHandler implements Runnable {
    //Clase que maneja la comunicacion con varios clientes a la vez
    private Socket clienteS;
    DataOutputStream out;
    DataInputStream in;

    public clienteHandler(Socket clienteS) {
        this.clienteS = clienteS;
    }

    @Override
    public void run() {
        try {
            //Creamos un login basico
            out = new DataOutputStream(out:clienteS.getOutputStream());
            out.writeUTF(str:"Introduzca usuario:");
            in = new DataInputStream(in: clienteS.getInputStream());

            out.writeUTF(str:"Introduzca password:");

            if (in.readUTF().equals(anObject:"roy") && in.readUTF().equals(anObject:"root")) {
                int estado = 1;
                //En caso de acertar entramos al programa
                do {
                    switch (estado) {
                        case 1:

                            out = new DataOutputStream(out:clienteS.getOutputStream());
                            out.writeUTF("Bienvenido! \n "
                                    + "Elige una de las siguientes opciones: \n"
                                    + "*Ver directorio actual \n"
                                    + "*Ver un archivo determinado \n"
                                    + "*Salir");
                            in = new DataInputStream(in: clienteS.getInputStream());
                            String respuesta = in.readUTF();
                            if (respuesta.equals(anObject:"Ver directorio actual")) {
                                String mensaje = "Contenido del directorio actual:";
                                String directorioActual = System.getProperty(key:"user.dir");
                                File directorio = new File(pathname:directorioActual);
                                String[] contenido = directorio.list();
                                if (contenido != null) {
                                    System.out.println(x: "Contenido del directorio actual:");
                                    for (String elemento : contenido) {
                                        System.out.println(x: elemento);
                                        mensaje += elemento + "\n";
```

En nuestro cliente handler establecemos el login con nombre de usuario y password, una vez este se loguea correctamente puede visualizar el contenido del directorio actual (user.dir), el diagrama de funciones esta en el mismo método run de la clase clienteHandler