**Actividad**

**1 Obligatoria Segundo Trimestre**

**Ivan Nuñez Rodriguez**

2º DAM

Programación de Procesos y Servicios

**Índice**

1. Código.
2. Resultado.
3. Enlace de Github

**Código**

**CLIENTE**

**ClienteVideoclub**

import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.PrintWriter;  
import java.net.Socket;  
import java.util.Scanner;  
  
class ClienteVideoclub {  
 public static void main(String[] args) {  
 try (Socket cliente = new Socket("localhost", 2001)) {  
 BufferedReader reader = new BufferedReader(new InputStreamReader(cliente.getInputStream()));  
 PrintWriter writer = new PrintWriter(cliente.getOutputStream(), true);  
 Scanner scanner = new Scanner(System.*in*);  
  
 String respuesta;  
 while ((respuesta = reader.readLine()) != null) {  
 System.*out*.println(respuesta);  
 String comando = scanner.nextLine();  
 writer.println(comando);  
 writer.flush();  
  
 if (comando.equals("1") || comando.equals("2")) {  
 respuesta = reader.readLine();  
 System.*out*.println(respuesta);  
 String parametro = scanner.nextLine();  
 writer.println(parametro);  
 writer.flush();  
 }  
  
 if (comando.equals("3")) {  
 break;  
 }  
 }  
 scanner.close();  
 } catch (IOException e) {  
 System.*out*.println("Error: " + e.getMessage());  
 }  
 }  
}

**SERVIDOR**

**HiloVideoCLub**

import com.google.gson.Gson;  
  
import java.io.\*;  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
class HiloVideoclub implements Runnable {  
 private static List<Pelicula> *peliculas* = *cargarPeliculas*();  
 private static int *contadorClientes* = 0;  
 private Socket enchufeAlCliente;  
 private Thread hilo;  
  
 public HiloVideoclub(Socket cliente) {  
 this.enchufeAlCliente = cliente;  
 *contadorClientes*++;  
 hilo = new Thread(this, "Cliente-" + *contadorClientes*);  
 hilo.start();  
 }  
  
 private static List<Pelicula> cargarPeliculas() {  
 List<Pelicula> peliculas = new ArrayList<>();  
 try (Reader reader = new FileReader("src/main/resources/peliculas.json")) {  
 Gson gson = new Gson();  
 Pelicula[] arrayPeliculas = gson.fromJson(reader, Pelicula[].class);  
 peliculas = new ArrayList<>(Arrays.*asList*(arrayPeliculas));  
 } catch (IOException e) {  
 System.*out*.println("Error al cargar las películas: " + e.getMessage());  
 e.printStackTrace();  
 }  
 return peliculas;  
 }  
  
 private static synchronized void modificarPelicula(int id, int opcionCampo, String nuevoValor, PrintWriter writer) {  
 for (Pelicula pelicula : *peliculas*) {  
 if (pelicula.getId() == id) {  
 switch (opcionCampo) {  
 case 1:  
 try {  
 pelicula.setRating(Double.*parseDouble*(nuevoValor));  
 } catch (NumberFormatException e) {  
 writer.println("Error: Valor de rating inválido.");  
 return;  
 }  
 break;  
 case 2:  
 pelicula.setComentarios(nuevoValor);  
 break;  
 default:  
 writer.println("Error: Opción no válida. Usa 1 para rating o 2 para comentarios.");  
 return;  
 }  
 writer.println("Película actualizada correctamente. " + pelicula);  
 return;  
 }  
 }  
 writer.println("Error: No se encontró ninguna película con el ID proporcionado.");  
  
 }  
  
 @Override  
 public void run() {  
 try (  
 InputStream entrada = enchufeAlCliente.getInputStream();  
 OutputStream salida = enchufeAlCliente.getOutputStream();  
 BufferedReader reader = new BufferedReader(new InputStreamReader(entrada));  
 PrintWriter writer = new PrintWriter(salida, true)  
 ) {  
 writer.println("Bienvenido al videoclub. Seleccione una opción: 1 - CONSULTA 2 - MODIFICAR 3 - FIN");  
  
  
 String comando;  
 while ((comando = reader.readLine()) != null) {  
 switch (comando) {  
 case "3":  
 writer.println("Gracias por usar el videoclub. ¡Hasta pronto!");  
 return;  
 case "1":  
 writer.println("Seleccione el tipo de consulta: 1 - ID 2 - TÍTULO");  
 String tipoConsulta = reader.readLine();  
  
 if (tipoConsulta.equals("1")) {  
 writer.println("Introduce el ID de la película:");  
 try {  
 int id = Integer.*parseInt*(reader.readLine());  
 List<Pelicula> encontradas = new ArrayList<>();  
 for (Pelicula p : *peliculas*) {  
 if (p.getId() == id) {  
 encontradas.add(p);  
 }  
 }  
 if (!encontradas.isEmpty()) {  
 for (Pelicula p : encontradas) {  
 writer.println(p);  
 }  
 } else {  
 writer.println("No se encontraron películas con el ID: " + id);  
 }  
  
  
 } catch (NumberFormatException e) {  
 writer.println("Error: ID debe ser un número entero.");  
 }  
 } else if (tipoConsulta.equals("2")) {  
 writer.println("Introduce el título de la película:");  
 String titulo = reader.readLine().trim().toLowerCase();  
 List<Pelicula> encontradas = new ArrayList<>();  
 for (Pelicula p : *peliculas*) {  
 if (p.getTitulo().toLowerCase().contains(titulo)) {  
 encontradas.add(p);  
 }  
 }  
 if (!encontradas.isEmpty()) {  
 for (Pelicula p : encontradas) {  
 writer.println(p);  
 }  
 } else {  
 writer.println("No se encontraron películas con título que contenga: " + titulo);  
 }  
  
 } else {  
 writer.println("Opción no válida. Usa 1 para ID o 2 para TÍTULO.");  
  
 }  
 break;  
 case "2":  
 writer.println("Introduce el ID de la película:");  
 int id = Integer.*parseInt*(reader.readLine());  
 writer.println("Seleccione el campo a modificar: 1 - RATING 2 - COMENTARIOS");  
 int campo = Integer.*parseInt*(reader.readLine());  
 writer.println("Introduce el nuevo valor:");  
 String nuevoValor = reader.readLine();  
 *modificarPelicula*(id, campo, nuevoValor, writer);  
 break;  
 default:  
 writer.println("Comando no reconocido. Usa 1 para CONSULTA, 2 para MODIFICAR o 3 para FIN.");  
 }  
 }  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 }  
}

**Película**import lombok.AllArgsConstructor;  
import lombok.Data;  
  
@Data  
@AllArgsConstructor  
class Pelicula {  
 private int id;  
 private String titulo;  
 private String creator;  
 private double rating;  
 private String dates;  
 private String comentarios;  
}

**ServidorVideoclub**

import java.io.IOException;  
import java.net.InetSocketAddress;  
import java.net.ServerSocket;  
import java.net.Socket;  
  
public class ServidorVideoclub {  
 public static void main(String[] args) {  
 int numeroClientes = 0;  
 System.*out*.println("SERVIDOR DEL VIDEOCLUB");  
  
 try (ServerSocket servidor = new ServerSocket()) {  
 InetSocketAddress direccion = new InetSocketAddress("localhost", 2001);  
 servidor.bind(direccion);  
 System.*out*.println("Servidor listo en: " + direccion.getAddress());  
  
 while (true) {  
 Socket enchufeAlCliente = servidor.accept();  
 numeroClientes++;  
 System.*out*.println("Nueva conexión de cliente" + numeroClientes);  
 new HiloVideoclub(enchufeAlCliente);  
 }  
 } catch (IOException e) {  
 System.*out*.println("Error en el servidor: " + e.getMessage());  
 }  
 }  
}

**peluculas.json**

[  
 {  
 "id": 1,  
 "titulo": "Juego de Tronos",  
 "creator": "David Benioff, D.B. Weiss",  
 "rating": 9.2,  
 "dates": "2011-2019",  
 "comentarios": "HBO"  
 },  
 {  
 "id": 2,  
 "titulo": "Breaking Bad",  
 "creator": "Vince Gilligan",  
 "rating": 9.4,  
 "dates": "2008-2013",  
 "comentarios": "AMC"  
 },  
 {  
 "id": 3,  
 "titulo": "Stranger Things",  
 "creator": "Matt Duffer, Ross Duffer",  
 "rating": 8.7,  
 "dates": "2016-",  
 "comentarios": "Netflix"  
 },  
 {  
 "id": 4,  
 "titulo": "The Boys",  
 "creator": "Eric Kripke",  
 "rating": 8.7,  
 "dates": "2019-",  
 "comentarios": "Prime Video"  
 },  
 {  
 "id": 5,  
 "titulo": "The Mandalorian",  
 "creator": "Jon Favreau",  
 "rating": 8.8,  
 "dates": "2019-",  
 "comentarios": "Disney +"  
 },  
 {  
 "id": 6,  
 "titulo": "WandaVision",  
 "creator": "Jac Schaeffer",  
 "rating": 8,  
 "dates": "2021",  
 "comentarios": "Disney +"  
 },  
 {  
 "id": 7,  
 "titulo": "Peaky Blinders",  
 "creator": "Steven Knight",  
 "rating": 8.8,  
 "dates": "2013-",  
 "comentarios": "BBC"  
 },  
 {  
 "id": 8,  
 "titulo": "Vikings",  
 "creator": "Michael Hirst",  
 "rating": 8.5,  
 "dates": "2013-2020",  
 "comentarios": "Prime Video"  
 }  
]

**Ejecución**

**CLIENTE**

****

**Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Imagen que contiene computadora, monitor, computer, tabla

El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

**SERVIDOR**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Enalce a Github**

<https://github.com/Ivannunezrodriguez/Programacion_de_Servicios_y_Procesos_24-25/tree/d239ea90c2113d9bed570dfaebe917c94c9c673d/Actividades/tema_2/Actividad%20UF2%20Obligatoria>