PRÁCTICA 1: DAWFLIX 2.0. Ampliación del proyecto original

Modalidad: Por parejas

RAs evaluados:

- **RA5: Entrada y salida de información**
- **RA6: Tipos avanzados de datos: SET y MAP**
- **RA8: Persistencia de información en bases de datos**
- **RA9: Gestión de información en bases de datos**

Puntuación total: 10 puntos

Esta primera práctica está dirigida a alumnos que ya han realizado el proyecto **DAWFLIX**. El objetivo es ampliar progresivamente su funcionalidad incorporando el uso de estructuras de datos más avanzadas, manejo de ficheros y conexión con una base de datos **MySQL**.

Entregables

- Código fuente Java completo.
- Ficheros `.txt` y binarios generados.
- Script `.sql` con creación de base de datos y tablas.
- README`.md` con instrucciones para compilar y ejecutar el proyecto.

Requisitos y Evaluación (ordenados por UT)

Nº	Requisito	RA	UT	Puntos
1	Usar `HashSet` para evitar duplicados en la lista de favoritos de los usuarios.	RA6	UT9	1.0
2	Implementar `Map<Usuario, Map<Contenido, Integer>>` para registrar reproducciones.	RA6	UT9	1.0
3	Implementar un `TreeMap` que almacene contenidos ordenados por su orden natural (`Comparable`).	RA6	UT9	1.0
4	Guardar y cargar listas de favoritos y reproducciones usando ficheros de texto (`java.nio.file.Files`).	RA5	UT11	1.0
5	Almacenar y reproducir contenidos musicales desde ficheros binarios (`.wav`) usando `Clip`.	RA5	UT11	2.0
6	Configurar la conexión de Java con MySQL: incluir el conector JDBC, crear clase de conexión y probar conexión.	RA8	UT12	1.0
7	Implementar operaciones CRUD en MySQL con usuarios y contenidos.	RA9	UT12	2.0
8	Crear una consulta SQL en Java que muestre los contenidos más reproducidos por usuario.	RA9	UT12	1.0

PISTAS: Lectura y escritura de ficheros binarios con `.wav` y reproducción de música

Estas operaciones permiten guardar de forma binaria información relacionada con archivos de música (`.wav`) y reproducirla desde Java. Todas las clases utilizadas forman parte del JDK estándar, por lo que **no es necesario instalar librerías externas**.

---

1. Escritura de un listado de archivos `.wav` en un fichero binario

```
import java.io.*;
import java.nio.file.*;
import java.util.*;

public class GuardarCanciones {
    public static void main(String[] args) {
        List<String> canciones = List.of("audio/intro.wav", "audio/accion.wav");

        Path rutaFichero = Paths.get("data/canciones.dat");
        try (ObjectOutputStream oos = new
ObjectOutputStream(Files.newOutputStream(rutaFichero))) {
            oos.writeObject(canciones);
            System.out.println("Canciones guardadas en fichero binario.");
        } catch (IOException e) {
            System.out.println("Error al guardar: " + e.getMessage());
        }
    }
}
```

Truco: Puedes usar rutas relativas como `"audio/intro.wav"` si los archivos `.wav` están en una carpeta dentro del proyecto.

---

2. Lectura del fichero binario y reproducción de las canciones

```
import java.io.*;
import java.nio.file.*;
import java.util.*;
import javax.sound.sampled.*;

public class ReproducirCanciones {
    public static void main(String[] args) {
        Path rutaFichero = Paths.get("data/canciones.dat");

        try (ObjectInputStream ois = new
ObjectInputStream(Files.newInputStream(rutaFichero))) {
            List<String> canciones = (List<String>) ois.readObject();

            for (String ruta : canciones) {
                System.out.println("Reproduciendo: " + ruta);
                reproducirWav(ruta);
            }

        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error al leer o reproducir: " + e.getMessage());
        }
    }

    public static void reproducirWav(String ruta) {
        try {
            File archivo = new File(ruta);
            AudioInputStream audioStream = AudioSystem.getAudioInputStream(archivo);
            Clip clip = AudioSystem.getClip();
            clip.open(audioStream);
            clip.start();
```

```
            Thread.sleep(clip.getMicrosecondLength() / 1000); // Espera a que termine
        } catch (Exception e) {
            System.out.println("Error al reproducir " + ruta + ": " + e.getMessage());
        }
    }
}
```

Estructura recomendada del proyecto

```
/DAWFLIX
├── /src
│   └── (clases Java)
├── /audio
│   └── intro.wav
│   └── accion.wav
├── /data
│   └── canciones.dat
```

Recomendaciones para los alumnos

- Asegúrate de que los archivos `.wav` están en el **formato correcto** (PCM sin compresión).
- Usa `ObjectOutputStream` para guardar estructuras como `List<String>`.
- Usa `Clip` de `javax.sound.sampled` para reproducir los sonidos.
- Trata de implementar **tratamiento de errores** con `try/catch` y muestra mensajes claros al usuario.
- Si usas rutas relativas, recuerda ejecutar el programa desde la raíz del proyecto.

Integración con MySQL desde Java – Guía práctica

1. Añadir el conector JDBC

- Descargar `mysql-connector-j-x.x.x.jar` desde https://dev.mysql.com/downloads/connector/j/
- Añadirlo al proyecto (Build Path o Librerías del IDE)

---

2. Crear clase de conexión

```
public class ConexionBD {
    private static final String URL = "jdbc:mysql://localhost:3306/dawflix";
    private static final String USUARIO = "root";
    private static final String CONTRASENA = "tu_contrasena";

    public static Connection conectar() {
        try {
            return DriverManager.getConnection(URL, USUARIO, CONTRASENA);
        } catch (SQLException e) {
            System.out.println("Error de conexión: " + e.getMessage());
            return null;
        }
    }
}
```

---

3. Ejecutar una consulta SELECT

```
try (Connection conn = ConexionBD.conectar()) {
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT nombre, COUNT(*) FROM reproducciones GROUP
BY nombre");

    while (rs.next()) {
        System.out.println("Usuario: " + rs.getString(1) + ", Reproducciones: " +
rs.getInt(2));
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```

PRÁCTICA 2: DAWFLIX Lite – Proyecto base con ampliaciones

Modalidad: Por parejas

RAs evaluados:

- **RA5: Entrada y salida de información**
- **RA6: Tipos avanzados de datos: SET y MAP**
- **RA8: Persistencia de información en bases de datos**
- **RA9: Gestión de información en bases de datos**

Puntuación total: 10 puntos

> Esta segunda práctica está dirigida a alumnos que **no han realizado DAWFLIX**. Se divide en dos fases: una implementación mínima de la base de la aplicación, seguida de una ampliación que incorpora funcionalidades más avanzadas.

Parte 1: Implementación básica (obligatoria, no puntuable)

1. Crear clase abstracta `Contenido` con atributos comunes (título, duración).
2. Subclases `Pelicula`, `Serie`, `Musica` con atributos propios.
3. Interfaz `Reproducible` con método `reproducir()`.
4. Clase `Usuario` con lista de favoritos (`ArrayList<Contenido>`).
5. Métodos para añadir contenido a favoritos y reproducir desde consola.

Parte 2: Requisitos de DAWFLIX Lite y evaluación (ordenados por UT)

Nº	Requisito	RA	UT	Puntos
1	Usar `HashSet` para evitar duplicados en la lista de favoritos.	RA6	UT9	1.0
2	Implementar `Map<Usuario, Map<Contenido, Integer>>` para registrar reproducciones.	RA6	UT9	1.0
3	Implementar `TreeMap` para almacenar contenidos por orden natural (`Comparable`).	RA6	UT9	1.0
4	Guardar y cargar datos usando ficheros de texto y `java.nio.file.Files`.	RA5	UT11	1.0
5	Almacenar y reproducir contenidos musicales desde ficheros binarios (`.wav`).	RA5	UT11	2.0
6	Configurar la conexión con MySQL desde Java (añadir conector, crear clase de conexión).	RA8	UT12	1.0
7	Implementar operaciones CRUD con MySQL para usuarios y contenidos.	RA9	UT12	2.0
8	Ejecutar una consulta SQL para mostrar contenidos más reproducidos por usuario.	RA9	UT12	1.0

Reproducción de música en Java e Integración con mySQL- Pistas

(Ver bloque `ReproductorMusica` en la práctica 1). (Ver sección completa en práctica 1: añadir conector JDBC, clase `ConexionBD`, ejemplo SELECT con `ResultSet`)

Evaluación de la práctica y test individual

Con el fin de garantizar que todos los alumnos implicados en las prácticas han participado activamente y han comprendido los conceptos trabajados, la evaluación final se compondrá de **dos partes complementarias**:

1. Desarrollo de la práctica en pareja (60% de la nota final)

La práctica debe entregarse en pareja, cumpliendo todos los requisitos funcionales y técnicos indicados en este documento. Será evaluada según los criterios de corrección, organización, funcionalidad y adecuación a los Resultados de Aprendizaje (RA) y Unidades de Trabajo (UT) correspondientes.

La formación en parejas es obligatoria, y las parejas deben ser formadas por miembros en las mismas condiciones de partida: alumnos que hayan entregado DAWFLIX deben emparejarse entre sí, y del mismo modo los que no lo hayan entregado.

2. Test individual de comprensión (40% de la nota final)

Una vez entregada la práctica, cada alumno deberá realizar **una prueba tipo test individual**, diseñada para verificar que conoce en profundidad:

- Las estructuras de datos utilizadas (Set, Map, TreeMap)
- Lectura y escritura de ficheros de texto y binarios
- Integración y consultas con base de datos MySQL
- Principales conceptos de las UT9, UT11 y UT12
- Teoría y/o aplicación práctica de los RAs: RA5, RA6, RA8 y RA9

El test constará de preguntas de tipo teórico, resolución de fragmentos de código y selección de la implementación más adecuada para un requisito dado.

Criterios de superación

Para superar la evaluación global de la práctica:

- Es **obligatorio obtener al menos un 4 sobre 10 (40%) en cada una de las partes** (práctica + test).
- La **nota final** se calculará como:

```
Nota final = (60% práctica) + (40% test individual)
```

Ejemplo: un alumno que obtenga un 9 en la práctica y un 5 en el test tendrá una nota final de **7,4 que se aplicará a todos sus RA's evaluados en la práctica.**

Objetivo pedagógico

Esta doble evaluación tiene como finalidad:

- **Fomentar el trabajo colaborativo efectivo** en equipo.
- **Asegurar la comprensión individual** de cada alumno sobre los contenidos técnicos aplicados.
- Evitar repartos desiguales de tareas en el desarrollo de la práctica.