Sistema de Gestión de Plataforma de Streaming Musical "SoundWave"

📖 Contexto del Proyecto

Has sido contratado/a como desarrollador/a backend para **"SoundWave"**, una startup tecnológica que está lanzando su propia plataforma de streaming musical para competir con Spotify, Apple Music y otras plataformas del mercado. Tu misión es diseñar e implementar el núcleo del sistema que gestionará usuarios, contenido multimedia, playlists, sistema de recomendaciones y monetización mediante anuncios.

La plataforma debe ser robusta, escalable y manejar correctamente situaciones excepcionales que puedan ocurrir durante su operación. El CEO de la empresa ha enfatizado la importancia de la experiencia del usuario y la estabilidad del sistema, por lo que el manejo de excepciones es crítico.

🎯 Objetivos

Implementar un sistema completo de streaming musical que demuestre dominio de:

- Herencia y polimorfismo
- Clases abstractas e interfaces
- Composición y agregación
- Colecciones (ArrayList, HashMap)
- Enumeraciones
- Manejo de excepciones personalizadas
- Diseño orientado a objetos

📋 Requisitos Funcionales Detallados

1. Gestión de Usuarios

Clase Abstracta: `Usuario`

Debe contener:

Atributos protegidos:

- `id` (String): Identificador único del usuario
- `nombre` (String): Nombre completo

- `email` (String): Correo electrónico (debe validarse formato)
- `password` (String): Contraseña (mínimo 8 caracteres)
- `suscripcion` (TipoSuscripcion): Tipo de suscripción activa
- `misPlaylists` (ArrayList<Playlist>): Playlists creadas por el usuario
- `historial` (ArrayList<Contenido>): Historial de reproducción
- `fechaRegistro` (Date): Fecha de registro en la plataforma

Métodos abstractos:

- `reproducir(Contenido contenido)`: Cada tipo de usuario reproduce de forma diferente

Métodos concretos:

- `crearPlaylist(String nombre)`: Crea una nueva playlist personal
- `seguirPlaylist(Playlist playlist)`: Sigue una playlist pública
- `darLike(Contenido contenido)`: Da "me gusta" a un contenido
- `validarEmail()`: Valida formato de email (debe contener @ y dominio)
- `validarPassword()`: Valida que la contraseña cumpla requisitos
- `agregarAlHistorial(Contenido contenido)`: Agrega contenido al historial

Excepciones a lanzar:

- `EmailInvalidoException`: Si el email no tiene formato válido
- `PasswordDebilException`: Si la contraseña tiene menos de 8 caracteres

Clase Concreta: `UsuarioPremium`

Hereda de `Usuario` y añade:

Atributos privados:

- `descargasOffline` (boolean): Si tiene descargas activas
- `maxDescargas` (int): Número máximo de descargas (100)
- `descargados` (ArrayList<Contenido>): Lista de contenido descargado
- `calidadAudio` (String): Calidad de audio ("HIGH", "ULTRA")

Métodos:

- `reproducir(Contenido contenido)`: Reproduce sin restricciones ni anuncios
- `descargar(Contenido contenido)`: Descarga contenido para escucha offline
- `eliminarDescarga(Contenido contenido)`: Elimina una descarga
- `verificarEspacioDescarga()`: Verifica si hay espacio para descargas

Excepciones a lanzar:

- `LimiteDescargasException`: Si se alcanzó el límite de 100 descargas
- `ContenidoYaDescargadoException`: Si el contenido ya está descargado

Clase Concreta: `UsuarioGratuito`

Hereda de `Usuario` y añade:

Atributos privados:

- `anunciosEscuchados` (int): Contador de anuncios vistos
- `ultimoAnuncio` (Date): Timestamp del último anuncio
- `reproduccionesHoy` (int): Reproducciones en el día actual
- `limiteReproducciones` (int): Límite diario de 50 reproducciones

Métodos:

- `reproducir(Contenido contenido)`: Reproduce con anuncios cada 3 canciones
- `verAnuncio()`: Muestra un anuncio aleatorio
- `puedeReproducir()`: Verifica si puede reproducir (límite diario)
- `reiniciarContadorDiario()`: Reinicia el contador de reproducciones

Excepciones a lanzar:

- `LimiteDiarioAlcanzadoException`: Si alcanzó las 50 reproducciones diarias
- `AnuncioRequeridoException`: Si debe ver un anuncio antes de continuar

2. Contenido Multimedia

Clase Abstracta: `Contenido`

Atributos protegidos:

- `id` (String): Identificador único
- `titulo` (String): Título del contenido
- `reproducciones` (int): Contador de reproducciones
- `likes` (int): Contador de "me gusta"
- `duracionSegundos` (int): Duración en segundos
- `tags` (ArrayList<String>): Etiquetas para búsqueda
- `disponible` (boolean): Si está disponible para reproducción
- `fechaPublicacion` (Date): Fecha de publicación

Métodos abstractos:

- `reproducir()`: Lógica específica de reproducción

Métodos concretos:

- `aumentarReproducciones()`: Incrementa contador
- `agregarLike()`: Incrementa likes
- `esPopular()`: Retorna true si tiene más de 100,000 reproducciones
- `validarDuracion()`: Valida que la duración sea positiva

Excepciones a lanzar:

- `ContenidoNoDisponibleException`: Si el contenido no está disponible
- `DuracionInvalidaException`: Si la duración es <= 0

Clase Concreta: `Cancion`

Hereda de `Contenido` e implementa `Reproducible` y `Descargable`:

Atributos privados:

- `letra` (String): Letra completa de la canción
- `artista` (Artista): Referencia al artista (agregación)
- `album` (Album): Referencia al álbum (agregación)
- `genero` (GeneroMusical): Género de la canción
- `audioURL` (String): URL del archivo de audio

- `explicit` (boolean): Si contiene contenido explícito
- `ISRC` (String): Código internacional de grabación

Métodos:

- `reproducir()`: Implementa la reproducción de la canción
- `obtenerLetra()`: Retorna la letra
- `esExplicit()`: Retorna si es contenido explícito
- `cambiarGenero(GeneroMusical nuevoGenero)`: Cambia el género

Excepciones a lanzar:

- `LetraNoDisponibleException`: Si no hay letra disponible
- `ArchivoAudioNoEncontradoException`: Si la URL del audio es inválida

Clase Concreta: `Podcast`

Hereda de `Contenido` e implementa `Reproducible` y `Descargable`:

Atributos privados:

- `creador` (Creador): Creador del podcast (agregación)
- `numeroEpisodio` (int): Número de episodio
- `temporada` (int): Temporada a la que pertenece
- `descripcion` (String): Descripción del episodio
- `categoria` (CategoriaPodcast): Categoría
- `invitados` (ArrayList<String>): Lista de invitados
- `transcripcion` (String): Transcripción del episodio

Métodos:

- `reproducir()`: Implementa la reproducción del podcast
- `obtenerDescripcion()`: Retorna descripción
- `agregarInvitado(String nombre)`: Agrega un invitado
- `esTemporadaNueva()`: Verifica si es de la última temporada

Excepciones a lanzar:

- `EpisodioNoEncontradoException`: Si el episodio no existe
- `TranscripcionNoDisponibleException`: Si no hay transcripción

3. Artistas, Creadores y Álbumes

Clase: `Artista`

Atributos privados:

- `id` (String): Identificador único
- `nombreArtistico` (String): Nombre artístico
- `nombreReal` (String): Nombre real
- `paisOrigen` (String): País de origen
- `discografia` (ArrayList<Cancion>): Todas sus canciones
- `albumes` (ArrayList<Album>): Sus álbumes
- `oyentesMensuales` (int): Oyentes mensuales
- `verificado` (boolean): Si está verificado en la plataforma
- `biografia` (String): Biografía del artista

Métodos:

- `publicarCancion(Cancion cancion)`: Publica nueva canción
- `crearAlbum(String titulo, Date fecha)`: Crea nuevo álbum
- `obtenerTopCanciones(int cantidad)`: Retorna las más populares
- `calcularPromedioReproducciones()`: Calcula promedio de todas sus canciones
- `esVerificado()`: Retorna si está verificado

Excepciones a lanzar:

- `ArtistaNoVerificadoException`: Si intenta realizar acciones de artista verificado
- `AlbumYaExisteException`: Si intenta crear un álbum con nombre duplicado

Clase: `Album`

Relación de COMPOSICIÓN con Cancion (si se elimina el álbum, se eliminan sus canciones del contexto del álbum)

Atributos privados:

- `id` (String): Identificador único
- `titulo` (String): Título del álbum
- `artista` (Artista): Artista principal
- `fechaLanzamiento` (Date): Fecha de lanzamiento

- `canciones` (ArrayList<Cancion>): Canciones del álbum (composición)
- `portadaURL` (String): URL de la portada
- `discografica` (String): Sello discográfico
- `tipoAlbum` (String): "LP", "EP", "Single"

Métodos:

- `crearCancion(String titulo, int duracion, GeneroMusical genero)`: **Crea una canción (COMPOSICIÓN)**
- `agregarCancion(Cancion cancion)`: Agrega canción al álbum
- `eliminarCancion(int posicion)`: Elimina canción por posición
- `getDuracionTotal()`: Calcula duración total
- `getNumCanciones()`: Retorna cantidad de canciones
- `ordenarPorPopularidad()`: Ordena canciones por reproducciones

Excepciones a lanzar:

- `AlbumCompletoException`: Si el álbum ya tiene el máximo de canciones (20)
- `CancionNoEncontradaException`: Si intenta eliminar una canción inexistente

Clase: `Creador`

Atributos privados:

- `id` (String): Identificador único
- `nombreCanal` (String): Nombre del canal/programa
- `nombre` (String): Nombre del creador
- `episodios` (ArrayList<Podcast>): Episodios publicados
- `suscriptores` (int): Número de suscriptores
- `descripcion` (String): Descripción del canal
- `redesSociales` (HashMap<String, String>): Redes sociales
- `categoriasPrincipales` (ArrayList<CategoriaPodcast>): Categorías

Métodos:

- `publicarPodcast(Podcast episodio)`: Publica nuevo episodio
- `obtenerEstadisticas()`: Genera objeto EstadisticasCreador
- `agregarRedSocial(String red, String usuario)`: Agrega red social

- `calcularPromedioReproducciones()`: Calcula promedio
- `eliminarEpisodio(String idEpisodio)`: Elimina un episodio

Excepciones a lanzar:

- `LimiteEpisodiosException`: Si intenta publicar más de 500 episodios
- `EpisodioNoEncontradoException`: Si intenta eliminar episodio inexistente

4. Sistema de Playlists

Clase: `Playlist`

Relación de AGREGACIÓN con Contenido (los contenidos existen independientemente de la playlist)

Atributos privados:

- `id` (String): Identificador único
- `nombre` (String): Nombre de la playlist
- `creador` (Usuario): Usuario que la creó
- `contenidos` (ArrayList<Contenido>): Lista de contenidos
- `esPublica` (boolean): Si es visible para otros
- `seguidores` (int): Número de seguidores
- `descripcion` (String): Descripción
- `portadaURL` (String): URL de imagen de portada
- `fechaCreacion` (Date): Fecha de creación
- `maxContenidos` (int): Máximo 500 contenidos

Métodos:

- `agregarContenido(Contenido contenido)`: Agrega contenido
- `eliminarContenido(String idContenido)`: Elimina contenido
- `ordenarPor(CriterioOrden criterio)`: Ordena la playlist
- `getDuracionTotal()`: Calcula duración total
- `shuffle()`: Mezcla aleatoriamente el orden
- `buscarContenido(String termino)`: Busca en la playlist
- `hacerPublica()`: Cambia a pública
- `hacerPrivada()`: Cambia a privada

Excepciones a lanzar:

- `PlaylistLlenaException`: Si se alcanza el límite de 500 contenidos
- `ContenidoDuplicadoException`: Si el contenido ya existe en la playlist
- `PlaylistVaciaException`: Si se intenta ordenar una playlist vacía

5. Sistema de Recomendaciones

Interfaz: `Recomendador`

```
public interface Recomendador {
    ArrayList<Contenido> recomendar(Usuario usuario) throws
RecomendacionException;
    ArrayList<Contenido> obtenerSimilares(Contenido contenido)
throws RecomendacionException;
}
```

Clase: `RecomendadorIA`

Implementa la interfaz `Recomendador`:

Atributos privados:

- `matrizPreferencias` (HashMap<String, ArrayList<String>>): Mapeo usuario → géneros preferidos
- `historialCompleto` (HashMap<String, ArrayList<Contenido>>): Historial por usuario
- `algoritmo` (AlgoritmoRecomendacion): Algoritmo utilizado
- `umbralSimilitud` (double): Umbral mínimo para recomendación (0.6)

Métodos:

- `recomendar(Usuario usuario)`: Genera recomendaciones personalizadas
- `obtenerSimilares(Contenido contenido)`: Encuentra contenido similar
- `entrenarModelo(ArrayList<Usuario> usuarios)`: Entrena modelo con datos

- `calcularSimilitud(Usuario u1, Usuario u2)`: Calcula similitud entre usuarios
- `actualizarPreferencias(Usuario usuario)`: Actualiza matriz de preferencias
- `obtenerGenerosPopulares()`: Retorna géneros más escuchados

Excepciones a lanzar:

- `RecomendacionException`: Excepción general del sistema de recomendaciones
- `HistorialVacioException`: Si el usuario no tiene historial para recomendar
- `ModeloNoEntrenadoException`: Si se intenta recomendar sin entrenar el modelo

6. Plataforma Principal

Clase: `Plataforma` (Singleton)

Clase principal que gestiona todo el sistema implementando el **patrón Singleton**.

Atributos privados:

- `instancia` (static Plataforma): Única instancia de la plataforma (Singleton)
- `nombre` (String): Nombre de la plataforma
- `usuariosPremium` (ArrayList<UsuarioPremium>): Usuarios premium registrados
- `usuariosGratuitos` (ArrayList<UsuarioGratuito>): Usuarios gratuitos registrados
- `catalogo` (ArrayList<Contenido>): Todo el contenido disponible
- `canciones` (ArrayList<Cancion>): Canciones del catálogo
- `podcasts` (ArrayList<Podcast>): Podcasts del catálogo
- `playlistsPublicas` (ArrayList<Playlist>): Playlists públicas
- `artistas` (HashMap<String, Artista>): Artistas por ID
- `albumes` (ArrayList<Album>): Álbumes registrados
- `creadores` (HashMap<String, Creador>): Creadores por ID
- `anuncios` (ArrayList<Anuncio>): Pool de anuncios
- `recomendador` (RecomendadorIA): Sistema de recomendaciones

Métodos del patrón Singleton:

- `getInstancia(String nombre)`: Retorna la única instancia, creándola si no existe
- `getInstancia()`: Retorna la instancia usando nombre por defecto "SoundWave"
- `reiniciarInstancia()`: Reinicia la instancia (útil para pruebas)
- Constructor privado: Impide instanciación directa con `new`

Métodos de registro de usuarios:

- `registrarUsuarioPremium(String nombre, String email, String password)`: Registra usuario premium
- `registrarUsuarioGratuito(String nombre, String email, String password)`: Registra usuario gratuito
- `getUsuariosPremium()`: Retorna lista de usuarios premium
- `getUsuariosGratuitos()`: Retorna lista de usuarios gratuitos
- `getTodosLosUsuarios()`: Retorna todos los usuarios como ArrayList<Usuario>

Métodos de gestión de artistas:

- `registrarArtista(String nombreArtistico, String nombreReal, String paisOrigen, boolean verificado, String biografia)`: Registra nuevo artista
- `buscarArtista(String nombre)`: Busca artista por nombre artístico
- `getArtistas()`: Retorna colección de todos los artistas
- `getArtistasVerificados()`: Retorna solo artistas verificados
- `getArtistasNoVerificados()`: Retorna solo artistas no verificados

Métodos de gestión de contenido:

- `crearAlbum(Artista artista, String titulo, Date fecha)`: Crea nuevo álbum para un artista
- `crearCancion(String titulo, int duracion, Artista artista, GeneroMusical genero)`: Crea canción independiente
- `crearCancionEnAlbum(String titulo, int duracion, Artista artista, Album album, GeneroMusical genero)`: Crea canción dentro de un álbum
- `registrarCreador(String nombreCanal, String nombre, String descripcion)`: Registra creador de podcasts
- `crearPodcast(String titulo, int duracion, Creador creador, int temporada, int episodio, CategoriaPodcast categoria)`: Crea nuevo podcast
- `agregarContenidoCatalogo(Contenido contenido)`: Agrega contenido al catálogo general

- `getCatalogo()`: Retorna todo el catálogo
- `getCanciones()`: Retorna solo canciones
- `getPodcasts()`: Retorna solo podcasts
- `getAlbumes()`: Retorna todos los álbumes
- `getTodosLosCreadores()`: Retorna todos los creadores

Métodos de búsqueda:

- `buscarContenido(String termino)`: Busca en el catálogo por título
- `buscarPorGenero(GeneroMusical genero)`: Filtra canciones por género
- `buscarPorCategoria(CategoriaPodcast categoria)`: Filtra podcasts por categoría
- `obtenerTopContenidos(int cantidad)`: Retorna los más populares ordenados por reproducciones

Métodos de playlists:

- `crearPlaylistPublica(String nombre, Usuario creador)`: Crea playlist pública
- `getPlaylistsPublicas()`: Retorna playlists públicas

Métodos de anuncios:

- `agregarAnuncio(Anuncio anuncio)`: Agrega anuncio al pool
- `obtenerAnuncioAleatorio()`: Retorna un anuncio aleatorio del pool
- `getAnuncios()`: Retorna todos los anuncios

Métodos de estadísticas y utilidades:

- `obtenerEstadisticasGenerales()`: Genera reporte completo de la plataforma
- `getRecomendador()`: Retorna el sistema de recomendaciones
- `getNombre()`: Retorna el nombre de la plataforma

Excepciones a lanzar:

- `UsuarioYaExisteException`: Si el email ya está registrado
- `ContenidoNoEncontradoException`: Si no se encuentra contenido en búsqueda
- `ArtistaNoEncontradoException`: Si el artista no existe en la plataforma
- `EmailInvalidoException`: Si el email no tiene formato válido

- `PasswordDebilException`: Si la contraseña no cumple requisitos
- `ArtistaNoVerificadoException`: Si un artista no verificado intenta crear álbum

Características del patrón Singleton implementado:

- **Thread-safe**: Métodos `getInstancia()` sincronizados
- **Lazy initialization**: La instancia se crea solo cuando se solicita
- **Acceso global**: Cualquier clase puede obtener la instancia mediante `Plataforma.getInstancia()`

7. Sistema de Anuncios

Clase: `Anuncio`

Atributos privados:

- `id` (String): Identificador único
- `empresa` (String): Empresa anunciante
- `duracionSegundos` (int): Duración del anuncio
- `audioURL` (String): URL del audio del anuncio
- `tipo` (TipoAnuncio): Tipo de anuncio
- `impresiones` (int): Veces que se ha mostrado
- `presupuesto` (double): Presupuesto del anuncio
- `activo` (boolean): Si está activo

Métodos:

- `reproducir()`: Reproduce el anuncio
- `registrarImpresion()`: Incrementa contador
- `calcularCostoPorImpresion()`: Calcula CPM
- `desactivar()`: Desactiva el anuncio

8. Interfaces Obligatorias

Interfaz: `Reproducible`

```
public interface Reproducible {
    void play();
```

```
    void pause();
    void stop();
    int getDuracion();
}
```

Interfaz: Descargable

```
public interface Descargable {
    boolean descargar() throws LimiteDescargasException,
ContenidoYaDescargadoException;
    boolean eliminarDescarga();
    int espacioRequerido();
}
```

9. Enumeraciones Obligatorias

TipoSuscripcion

```
public enum TipoSuscripcion {
    GRATUITO(0.0, false, 50, false),
    PREMIUM(9.99, true, -1, true),
    FAMILIAR(14.99, true, -1, true),
    ESTUDIANTE(4.99, true, -1, true);

    private double precioMensual;
    private boolean sinAnuncios;
    private int limiteReproducciones;
    private boolean descargasOffline;

    // Constructor, getters y métodos
}
```

GeneroMusical

```
public enum GeneroMusical {
    POP, ROCK, HIPHOP, JAZZ, ELECTRONICA,
    REGGAETON, INDIE, CLASICA, COUNTRY,
    METAL, RNB, SOUL, BLUES, TRAP
}
```

CategoriaPodcast

```
public enum CategoriaPodcast {
    TECNOLOGIA, DEPORTES, COMEDIA, TRUE_CRIME,
    EDUCACION, NEGOCIOS, SALUD, ENTRETENIMIENTO,
    HISTORIA, CIENCIA, POLITICA, CULTURA
}
```

TipoAnuncio

```
public enum TipoAnuncio {
    AUDIO(15, 0.05),
    BANNER(0, 0.02),
    VIDEO(30, 0.10);

    private int duracionSegundos;
    private double costoPorImpresion;

    // Constructor y getters
}
```

CriterioOrden

```
public enum CriterioOrden {
    FECHA_AGREGADO,
    POPULARIDAD,
    DURACION,
    ALFABETICO,
    ARTISTA,
    ALEATORIO
}
```

AlgoritmoRecomendacion

```
public enum AlgoritmoRecomendacion {
    COLABORATIVO("Basado en usuarios similares"),
    CONTENIDO("Basado en características del contenido"),
    HIBRIDO("Combinación de ambos");

    private String descripcion;

    // Constructor y getters
}
```

10. Excepciones Personalizadas Obligatorias

Debes crear las siguientes excepciones personalizadas:

Excepciones de Usuario:

```
public class EmailInvalidoException extends Exception
public class PasswordDebilException extends Exception
public class LimiteDiarioAlcanzadoException extends Exception
public class AnuncioRequeridoException extends Exception
```

Excepciones de Contenido:

```
public class ContenidoNoDisponibleException extends Exception
public class DuracionInvalidaException extends Exception
public class LetraNoDisponibleException extends Exception
public class ArchivoAudioNoEncontradoException extends Exception
public class EpisodioNoEncontradoException extends Exception
public class TranscripcionNoDisponibleException extends Exception
```

Excepciones de Descargas:

```
public class LimiteDescargasException extends Exception
public class ContenidoYaDescargadoException extends Exception
```

Excepciones de Playlist:

```
public class PlaylistLlenaException extends Exception
public class ContenidoDuplicadoException extends Exception
public class PlaylistVaciaException extends Exception
public class CancionNoEncontradaException extends Exception
```

Excepciones de Artista/Creador:

```
public class ArtistaNoVerificadoException extends Exception
public class AlbumYaExisteException extends Exception
public class AlbumCompletoException extends Exception
public class LimiteEpisodiosException extends Exception
```

Excepciones de Plataforma:

```
public class UsuarioYaExisteException extends Exception
public class ContenidoNoEncontradoException extends Exception
public class ArtistaNoEncontradoException extends Exception
```

Excepciones de Recomendación:

```
public class RecomendacionException extends Exception
public class HistorialVacioException extends RecomendacionException
public class ModeloNoEntrenadoException extends
RecomendacionException
```

Todas las excepciones deben:

- Tener al menos 2 constructores (uno sin parámetros y otro con mensaje)
- Heredar de Exception

💻 Clase Main Obligatoria

Debes implementar un método `main` completo que demuestre **TODAS** las funcionalidades del sistema, incluyendo:

Escenario 1: Registro y Validación (5 puntos)

```
// 1. Crear la plataforma
// 2. Intentar registrar usuario con email inválido (capturar
EmailInvalidoException)
// 3. Intentar registrar usuario con password débil (capturar
PasswordDebilException)
// 4. Registrar correctamente 5 usuarios (3 premium, 2 gratuitos)
```

Escenario 2: Creación de Contenido (10 puntos)

```
// 1. Crear 3 artistas verificados
// 2. Crear 2 artistas no verificados
// 3. Crear 3 álbumes con 5-10 canciones cada uno
// 4. Intentar que artista no verificado cree álbum (capturar
ArtistaNoVerificadoException)
// 5. Crear 2 creadores de podcasts
// 6. Crear 10 episodios de podcast distribuidos entre los creadores
// 7. Agregar todo el contenido al catálogo de la plataforma
```

Escenario 3: Reproducción y Límites (15 puntos)

```
// 1. Usuario gratuito reproduce 3 canciones consecutivas
//    - Debe ver anuncio después de la 3ra canción (capturar
AnuncioRequeridoException)
// 2. Usuario gratuito intenta reproducir 51 canciones en un día
//    - Debe lanzar LimiteDiarioAlcanzadoException
// 3. Usuario premium reproduce 10 canciones sin interrupciones
// 4. Usuario premium descarga 5 canciones
// 5. Usuario premium intenta descargar la misma canción (capturar
ContenidoYaDescargadoException)
```

Escenario 4: Gestión de Playlists (10 puntos)

```
// 1. Usuario premium crea 2 playlists privadas
// 2. Usuario gratuito crea 1 playlist pública
// 3. Agregar 20 canciones a una playlist
// 4. Intentar agregar canción duplicada (capturar
ContenidoDuplicadoException)
// 5. Intentar agregar 501 canciones a una playlist (capturar
PlaylistLlenaException)
// 6. Ordenar playlist por popularidad
// 7. Ordenar playlist por duración
// 8. Hacer shuffle de una playlist
// 9. Usuario premium sigue playlist pública de usuario gratuito
```

Escenario 5: Búsquedas y Filtros (10 puntos)

```
// 1. Buscar contenido por término "love"
// 2. Buscar canciones del género REGGAETON
// 3. Buscar podcasts de categoría TECNOLOGIA
// 4. Buscar contenido inexistente (capturar
ContenidoNoEncontradoException)
// 5. Obtener top 10 contenidos más reproducidos
// 6. Buscar artista por nombre
// 7. Buscar artista inexistente (capturar
ArtistaNoEncontradoException)
```

Escenario 6: Sistema de Recomendaciones (10 puntos)

```
// 1. Entrenar el modelo de recomendaciones con todos los usuarios
// 2. Generar recomendaciones para usuario con historial amplio
// 3. Intentar generar recomendaciones para usuario sin historial
//    (capturar HistorialVacioException)
// 4. Obtener contenido similar a una canción popular
// 5. Obtener contenido similar a un podcast
```

Escenario 7: Gestión de Álbumes y Artistas (10 puntos)

```
// 1. Artista publica nuevo álbum
// 2. Intentar crear álbum con nombre duplicado (capturar
AlbumYaExisteException)
// 3. Agregar canciones al álbum hasta completar 20
// 4. Intentar agregar canción 21 (capturar AlbumCompletoException)
// 5. Eliminar una canción del álbum
// 6. Intentar eliminar canción inexistente (capturar
CancionNoEncontradaException)
// 7. Calcular duración total del álbum
// 8. Obtener top 5 canciones del artista
```

Escenario 8: Creadores y Podcasts (10 puntos)

```
// 1. Creador publica episodio nuevo
// 2. Agregar invitados al episodio
// 3. Obtener estadísticas del creador
// 4. Eliminar un episodio antiguo
// 5. Intentar eliminar episodio inexistente (capturar
EpisodioNoEncontradoException)
// 6. Calcular promedio de reproducciones del creador
```

Escenario 9: Contenido No Disponible (10 puntos)

```
// 1. Marcar un contenido como no disponible
// 2. Intentar reproducir contenido no disponible (capturar
ContenidoNoDisponibleException)
// 3. Buscar letra de canción sin letra (capturar
LetraNoDisponibleException)
// 4. Buscar transcripción de podcast sin transcripción
//    (capturar TranscripcionNoDisponibleException)
```

Escenario 10: Estadísticas Finales (10 puntos)

```
// 1. Mostrar total de usuarios registrados (por tipo)
// 2. Mostrar total de contenido en catálogo (canciones vs podcasts)
// 3. Mostrar artista con más reproducciones
// 4. Mostrar creador con más suscriptores
// 5. Mostrar playlist pública más seguida
// 6. Mostrar género musical más popular
// 7. Mostrar categoría de podcast más popular
// 8. Calcular ingresos totales de la plataforma (basado en
suscripciones)
// 9. Mostrar número total de anuncios reproducidos
// 10. Generar reporte completo de la plataforma
```

📊 Formato de Salida del Main

El programa debe mostrar salidas claras y organizadas. Ejemplo:

```
==========================================
     SOUNDWAVE - SISTEMA DE STREAMING
==========================================

[ESCENARIO 1: REGISTRO DE USUARIOS]
✓ Plataforma creada exitosamente
```

```
✗ Error: Email inválido - juan@gmail (EmailInvalidoException)
✗ Error: Contraseña débil - mínimo 8 caracteres
(PasswordDebilException)
✓ Usuario Premium registrado: Juan Pérez (juan@gmail.com)
✓ Usuario Premium registrado: María López (maria@hotmail.com)
✓ Usuario Gratuito registrado: Pedro García (pedro@yahoo.com)

[ESCENARIO 2: CREACIÓN DE CONTENIDO]
✓ Artista creado: Bad Bunny (verificado)
✓ Álbum creado: "Un Verano Sin Ti" - 23 canciones
✓ Canción agregada: "Moscow Mule" - Género: REGGAETON
✗ Error: Artista no verificado no puede crear álbum
(ArtistaNoVerificadoException)

[ESCENARIO 3: REPRODUCCIÓN]
♪ [Usuario Gratuito - Pedro] Reproduciendo: "Moscow Mule"
♪ [Usuario Gratuito - Pedro] Reproduciendo: "Tití Me Preguntó"
♪ [Usuario Gratuito - Pedro] Reproduciendo: "Party"
⚠ [ANUNCIO REQUERIDO] Viendo anuncio de Nike (15 segundos)
...

[ESTADÍSTICAS FINALES]
========================
Total Usuarios: 5 (3 Premium, 2 Gratuitos)
Total Contenido: 150 (120 canciones, 30 podcasts)
Artista más popular: Bad Bunny (5,000,000 reproducciones)
Género más escuchado: REGGAETON
Ingresos mensuales: $44.97
========================
```

🎓 Entregables

1. **Código Fuente Completo en GitHub**

⏱️ Tiempo Estimado

- **Lectura y planificación:** 2 horas
- **Implementación de clases base:** 10 sesiones
- **Implementación de excepciones:** 5 sesiomes
- **Implementación del Main:** 5 sesiones minutos

Estructura de Carpetas:

```
src/
├── Main.java
├── enums/
│   ├── TipoSuscripcion.java
│   ├── GeneroMusical.java
│   ├── CategoriaPodcast.java
│   ├── TipoAnuncio.java
│   ├── CriterioOrden.java
│   └── AlgoritmoRecomendacion.java
├── excepciones/
│   ├── usuario/
│   │   ├── EmailInvalidoException.java
│   │   ├── PasswordDebilException.java
│   │   ├── LimiteDiarioAlcanzadoException.java
│   │   └── AnuncioRequeridoException.java
│   ├── contenido/
│   │   ├── ContenidoNoDisponibleException.java
│   │   ├── DuracionInvalidaException.java
│   │   ├── LetraNoDisponibleException.java
│   │   ├── ArchivoAudioNoEncontradoException.java
│   │   ├── EpisodioNoEncontradoException.java
│   │   └── TranscripcionNoDisponibleException.java
│   ├── descarga
│   │   ├── LimiteDescargasException.java
│   │   └── ContenidoYaDescargadoException.java
│   ├── playlist/
│   │   ├── PlaylistLlenaException.java
│   │   ├── ContenidoDuplicadoException.java
│   │   ├── PlaylistVaciaException.java
│   │   └── CancionNoEncontradaException.java
│   ├── artista
│   │   ├── ArtistaNoVerificadoException.java
│   │   ├── AlbumYaExisteException.java
│   │   ├── AlbumCompletoException.java
│   │   └── LimiteEpisodiosException.java
│   ├── plataforma/
│   │   ├── UsuarioYaExisteException.java
│   │   ├── ContenidoNoEncontradoException.java
│   │   └── ArtistaNoEncontradoException.java
│   └── recomendación/
│       ├── RecomendacionException.java
│       ├── HistorialVacioException.java
│       └── ModeloNoEntrenadoException.java
├── interfaces/
│   ├── Reproducible.java
│   ├── Descargable.java
│   └── Recomendador.java
├── modelo/
│   ├── usuarios/
│   │   ├── Usuario.java
│   │   ├── UsuarioPremium.java
│   │   └── UsuarioGratuito.java
│   ├── contenido/
│   │   ├── Contenido.java
│   │   ├── Cancion.java
│   │   └── Podcast.java
```

```
│   ├── artistas/
│   │   ├── Artista.java
│   │   ├── Album.java
│   │   └── Creador.java
│   └── plataforma/
│       ├── Plataforma.java              # Clase principal
│       ├── Playlist.java
│       └── Anuncio.java
└── utilidades/
    ├── RecomendadorIA.java
    └── EstadisticasCreador.java
```

⚠️ Restricciones Importantes

1. **NO** uses librerías externas excepto java.util y java.io
2. **NO** uses bases de datos, todo debe ser en memoria
3. Debes validar todos los datos de entrada
4. Debes manejar TODAS las excepciones apropiadamente
5. Eprograma debe compilar sin errores
6. El main debe ejecutarse completamente sin crashes

🚀 Bonus (Puntos Extra - Opcional)

- **(+ 20 puntos)** Implementar patrón Singleton en la clase Plataforma

---

📘 Fecha de Entrega

[15/02/2026 – 23:59]

Formato: un txt con el enlace a tu github del proyecto.