Nombre y Apellido: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Cantidad de hojas entregadas: \_\_ Nota:\_\_\_\_\_\_\_\_

|  |
| --- |
| **Lea atentamente TODO el enunciado antes de empezar a desarrollar, recodar generar todas las firmas de las funciones y las especificaciones correspondientes para que el proyecto compile antes de programar la solución** |

1) Un Concierto es un lugar donde llegan bandas y allí reproducen su Musica. Se le pidió a Bárbara crear una clase que modele un concierto. A Bárbara se le ocurrió que para modelar la clase necesita solo un método independientemente de la banda y hacerlo funcionar de manera polimorfica. Para ello se cuenta con la interfaz Banda (5ptos)  
Se pide:  
a) Implementar la función en java dentro de la clase Concierto que tenga el método static String reproducirTodas() que recibe una lista con todas los bandas a reproducir y devuelve en un String el resultado de haber reproducido a todas. (El método debe reproducir a cualquier banda independientemente de quien sea). Si no hay bandas, debe lanzar la excepción NoHayBandasException.  
Pista: En cada elemento de la lista se encuentra un objeto que implementa a Banda. Pero al no ser una clase, nada es instancia de Banda  
AYUDA: reproducirTodas() todas no es mas que la concatenación de todos los Strings de las bandas reproducidas.

b) Implementar las clases ACDC y TheBeatles con UNICAMENTE la primitiva *String reproducir()* que permite que la función implementada en a) funcione.   
(Cuando se reproduce una banda esto no es mas que devolver un String que diga “Suena TheBeatles” o “Suena ACDC”)

AYUDA: Ambos constructores son por defecto y no setean ningún atributo.  
AYUDA2: Para evitar problemas de comparación, copiar los resultados de la reproducción que estan entre comillas.

2) Dado el código de ejemplo adjunto, se tiene un problema de diseño. Para la aplicación AccesoADatos solicitada, el método conectar obtiene los datos de una DataBase. Pero el cliente decidió un buen día que no era necesario utilizar los datos de una DataBase, sino que en algunos casos conviene obtenerlos de una API.   
El problema es que si quitamos la DataBase el programa pierde compatibilidad hacia atrás. Por otro lado, el día de mañana se sabe que la nueva tecnología llega por un nuevo CloudService que aún no salió al mercado. (5ptos)

a) Utilizando el quinto principio de SOLID, y con la abstracción que se brinda en la consigna se solicita cambiar la clase AccesoADatos para que continúe funcionando para la DataBaseService y ahora también para el APIService que fue solicitado por el cliente.   
La abstracción debe contener un método abrirConexion simplemente haga funcionar correspondientemente a la DataBase a su manera (conectar,consultar y procesar) a la API a su manera(solicitarAcceso ,comprobarServidor,accesoAServidorAPI), y otro metodo successful que devuelve si se ejecuto correctamente al servicio.

AYUDA: La forma de ejecución “correcta” del servicio DataBase esta actualmente en la clase AccesoADatos. La forma correcta del APIService es análogo al de la DataBaseService. El orden de ejecución del API service es el indicado entre parentesís anteriormente.

AYUDA2: Recordar que se puede declarar un tipo de objeto como la interfaz que se implementa.  
Un ejemplo de esto puede hacerse con el tp de Pokemon, donde:

HoOh h1 = new HoOh();

Tambien puede escribirse como

TipoFuego h1 = new HoOh();

**IMPORTANTE: Para que el examen compile en su totalidad, debe solucionarse el problema de compatibilidad, de este punto.**