

En la unidad 3 de nuestro curso hemos trabajado sobre la aplicación de mecanismos de reutilización como son la **herencia**, **polimorfismo** y el uso de **interfaces**. Estas técnicas enriquecen el comportamiento de las clases modeladas como parte de una solución en el diagrama de clases.

En la unidad 4, hemos aprendido a utilizar el concepto de **matriz** como elemento de modelaje, pudiendo así agrupar elementos del modelo de la solución en una estructura contenedora de dos dimensiones de tamaño fijo; y hemos empleado **estructuras contenedoras lineales de tamaño variable** como elementos de modelado que permiten manejar una secuencia de objetos.

Esta tarea integradora presenta una actividad en la cual se requiere aplicar todos los conocimientos adquiridos hasta el momento. Por tanto, esta tarea es un instrumento para verificar el cumplimiento de los objetivos que han sido planteados para las unidades 3 y 4 descritos en el programa del curso. Para llevar a cabo este ejercicio es necesario realizar las actividades listadas a continuación:

Actividades

Lleve a cabo las siguientes actividades de cada una de las etapas de desarrollo de software:

1. **Análisis del problema.** [Tabla de especificación y Lista de requerimientos funcionales](#) en el formato visto en la clase de ingeniería de software 1.
2. **Diseño de la solución.**
 - a. Elabore un **diagrama de clases** que modele la solución del problema de acuerdo con las buenas prácticas y los patrones de diseño revisados hasta el momento en el curso. Su diagrama debe incluir el paquete modelo y el de interfaz de usuario.
 - b. **Trazabilidad del Análisis al Diseño.** Una [tabla a tres columnas](#) en la que se relaciona cada requerimiento con el método o métodos que permiten satisfacer dicho requerimiento.
 - c. Recuerde que todos los artefactos generados en la fase de diseño deben ser en inglés.
3. **Implementación en Java.** Incluya en la implementación, los comentarios descriptivos sobre los atributos y métodos de cada clase. Recuerde que todos los artefactos generados en la fase de implementación deben ser en inglés.
4. **Documentación en JavaDoc** (Debe entregarse el JavaDoc generado y ubicarlo en la carpeta doc).
5. **Usar GitHub como repositorio de código fuente y documentación** utilizando la estructura de carpetas aprendida en clase. En la entrega se debe adjuntar un enlace URL al repositorio. Se debe poder identificar commits en diferentes fechas.
6. Elaborar **un video con la ejecución del programa**. En la entrega se debe adjuntar un enlace al video.
7. Subir a Intu todos los entregables.

Recuerde que puede encontrar la **Rúbrica de la tarea integradora** en el siguiente [enlace](#).

Nota:

- Usted debe entregar la URL de su repositorio GitHub donde se deben encontrar los archivos de codificación en sus respectivos paquetes.
- Tenga en cuenta que su repositorio GitHub debe presentar una estructura base como por ejemplo:

```
proyecto/  
  src/  
  bin/  
  doc/
```
- Dentro de los directorios `src/` y `bin/` estarán presentes estos directorios (representando cada uno de sus paquetes):

```
  ui/  
  model/
```
- El directorio `src` (source code) contiene sus clases `.java` dentro del directorio `ui/` y `model/`. Por otro lado, el directorio `bin` (binary files) contiene los archivos `.class` en el directorio `ui/` y `model/`. El directorio `doc` tendrá toda la documentación de análisis y diseño
- Su código debería compilar de acuerdo con lo explicado en la diapositiva 15 de esta presentación: <http://tinyurl.com/y3bd9bg2>

A continuación, encontrará un enunciado que narra de forma detallada la situación problemática que se espera usted solucione.

Enunciado

CLIENTE

Debido a su extensa experiencia como programador, usted ha sido contratado por **NeoTunes**, una empresa, de base danesa, para desarrollar un prototipo de software que les permita entrar a competir en el mercado del streaming de música y contenido de audio. El modelo de negocio de la empresa tiene dos focos, las suscripciones de usuarios y la venta de canciones a través de la plataforma. Contrario a otros competidores del mercado, en NeoTunes los usuarios pueden ser **realmente dueños de su catálogo de música**.

Los usuarios consumidores de la plataforma pueden ser de dos tipos: **estándar** y **premium**. El usuario Estándar tendrá acceso al catálogo de audio, podrá crear hasta 20 listas de reproducción, tendrá la oportunidad de comprar hasta 100 canciones y durante su uso de la plataforma, se reproducirán esporádicamente (cada 2 canciones o antes de cada podcast) anuncios publicitarios¹. Por su parte, el usuario Premium tendrá acceso a todo el catálogo de audio, podrá crear listas de reproducción ilimitadas y comprar ilimitadamente canciones.

¹ El prototipo debe contener los siguientes anuncios publicitarios:

- Nike - Just Do It.
- Coca-Cola - Open Happiness.
- M&Ms - Melts in Your Mouth, Not in Your Hands

Para realizar el **registro de usuarios** en la plataforma se necesitan los siguientes datos: **nickname**, **cédula** y **fecha de vinculación** (pueden usar librerías). Cuando los **usuarios consumidores** compren una **canción**, se debe almacenar la **fecha de la operación**. Adicionalmente, con el fin de **generar contenidos dirigidos**, **NeoTunes** desea saber de cada usuario consumidor para cada tipo de contenido, **el acumulado de tiempo reproducido, el género o categoría más escuchado y el artista y creador de contenido más escuchados**.

Los usuarios **productores**, estos pueden ser de dos tipos: **artistas** y **creadores de contenido**. De ambos se necesita registrar su **nombre**, **fecha de vinculación a la plataforma** (pueden usar librerías) y **una URL con su foto o imagen distintiva**. Asimismo, se desea saber el **número acumulado de reproducciones** y **el total de tiempo reproducido por los usuarios consumidores**.

NeoTunes anticipa la evolución y crecimiento dinámico de su plataforma por lo que le solicita que el diseño del prototipo contemple la creación futura de otros tipos de usuario, tanto consumidores como productores.



Los **audios** que producen los artistas se denominan **canciones**. Cada canción tiene un **nombre**, un **álbum**, un **género**, una URL con la carátula del álbum al que pertenece, una **duración**, el **valor de venta** (en dólares), **el número de reproducciones** y **el número de veces vendida**. Los posibles géneros existentes en el prototipo son: **Rock, Pop, Trap y House**.

Por su parte, los **audios** producidos por los creadores de contenido se denominan **podcasts**. Cada podcast tiene un **nombre**, una **descripción**, una **categoría**, una **URL con la imagen distintiva** del mismo, una **duración** y **el número de reproducciones**. Los podcasts no se pueden comprar. Las posibles categorías existentes en el prototipo son: **Política, Entretenimiento, Videojuegos y Moda**.

NeoTunes anticipa cambios en el futuro de la industria de contenido por lo que le solicita que el diseño del prototipo contemple la creación futura de otros tipos de audio.

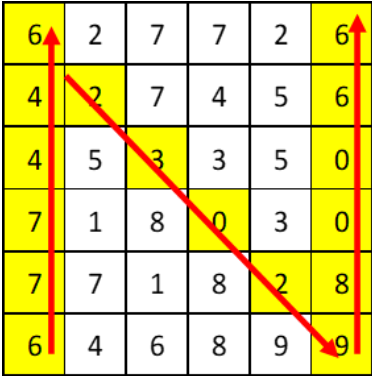
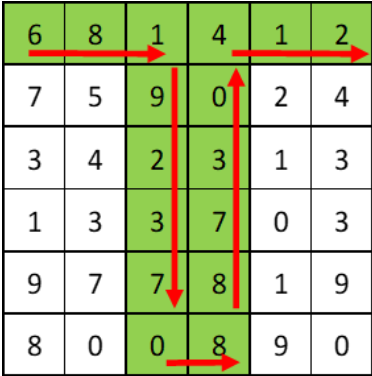
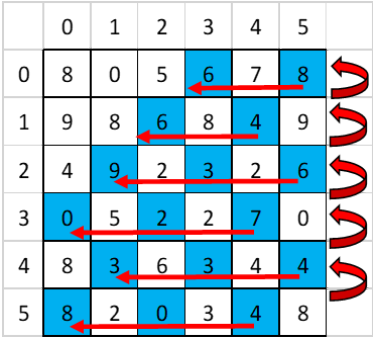
Ejemplo de Canción		Ejemplo de Podcast	
			
Nombre:	Numb	Name:	Episode 103: M&D
Álbum:	Meteora	Descripción:	New mini-set review
Género:	Rock	Categoría:	Videojuegos
URL:	LP_M.png	URL:	VS_DRP.jpeg

Ejemplo de Canción		Ejemplo de Podcast	
Duración:	3:05	Duración:	1:28:23
Valor de venta:	\$ 0,99 USD		
# de reproducciones:	1.112.018.485	# de reproducciones:	121.867
# de ventas:	13.300.000		

Por otro lado, las listas de reproducción son colecciones de archivos de audio. Cada lista de reproducción tiene un nombre, una lista de audios, ya sean canciones o podcasts y un código numérico autogenerated que servirá como identificador de la lista. La idea es que los usuarios puedan compartir el código y acceder directamente al contenido de la lista de reproducción de otros usuarios. El código se deberá generar como una cadena formada por 16 enteros que se encuentren en un **RECORRIDO** específico sobre una matriz (6X6) aleatoria de números (entre 0 y 9) de la siguiente manera:

1. Una lista de reproducción con solo Canciones: Recorrido en letra N
2. Una lista de reproducción con solo Podcasts: Recorrido en letra T
3. Una lista de reproducción con Canciones y Podcasts: el número de las casillas i,j, recorriendo la matriz de abajo hacia arriba y de derecha a izquierda, cuando la suma $i+j$ sea un número impar mayor que 1.

Se deberá mostrar, a través de la interfaz de usuario, tanto la matriz como el código identificador resultante, cada vez que el usuario decida compartir la lista.

Solo Canciones	Solo Podcast	Canciones y Podcasts
 <p>Código: 6774462302980066</p>	 <p>Código: 6819237088730412</p>	 <p>Código: 4084337206394686</p>

El prototipo a desarrollar debe ofrecer las siguientes funcionalidades:

1. Registrar usuarios productores, artistas y creadores de contenido.
2. Registrar usuarios consumidores, estándar y premium.
3. Registrar canciones y podcasts.
4. Crear una lista de reproducción.
5. Editar una lista de reproducción.
6. Compartir una lista de reproducción.
7. Simular la reproducción de una canción o podcast (estándar y premium).
8. Comprar una canción.
9. Generar informes con los datos registrados:
 - a. Para cada tipo de audio, canciones y podcast, informar el acumulado total de reproducciones en toda la plataforma.
 - b. Informar el género de canción más escuchado (nombre y número de reproducciones) para un usuario específico y para toda la plataforma y su número de reproducciones.
 - c. Informar la categoría de podcast más escuchada (nombre y número de reproducciones) para un usuario específico y para toda la plataforma.
 - d. De cada uno de los integrantes del Top 5 de artistas y del Top 5 de creadores de contenido en la plataforma, informar el nombre y número de reproducciones totales.
 - e. De cada uno de los integrantes del Top 10 de canciones y del Top 10 de podcast, informar el nombre, género o categoría y número total de reproducciones.
 - f. De cada género, informar el número de canciones vendidas y el valor total de ventas (\$).
 - g. De la canción más vendida en la plataforma, informar el número total de ventas y el valor total de venta (\$).

Entregas

Primera entrega

Se deben entregar la primera versión del diagrama de clases completo y todos los artefactos de análisis, diseño e implementación que le permitan completar los requerimientos de 1 a 5.

Segunda entrega

Se deben entregar la versión final del diagrama de clases completo y los artefactos de análisis, diseño e implementación que le permitan completar los requerimientos de 6 a 9.

Sustentación

Se realizará una sustentación individual ante los profesores de Algoritmos y Programación I e Ingeniería de Software I. Las fechas y lugares se definirán en el transcurso del semestre.