FUNDAMENTOS DE LA PROGRAMACIÓN – PROGRAMACIÓN ORIENTADA A OBJETOS

Proyecto: 500 mejores álbumes de la historia

Autor: Daniel Mateos
Revisor: Jose Antonio Troyano
Última modificación: 01/05/2019

1 Material

Para la realización de este ejercicio se dispone de los siguientes elementos contenidos en el fichero zip:

- **/doc/Enunciado.pdf**: fichero PDF con este enunciado
- **/data/**: carpeta de datos
  - **/data/500mejores.csv**: fichero CSV con los datos
- **/src/fp.albumes**: paquete Java con parte de la implementación de algunas clases del modelo
- **/src/fp.albumes.test**: paquete Java con las clases de test para las distintas clases que habrá que desarrollar en el proyecto
- **/src/fp.utiles**: paquete Java con utilidades de la asignatura

2 Datos disponibles

En este proyecto trabajaremos con datos sobre los mejores 500 álbumes de la historia según la revista Rolling Stone. En estos datos encontramos solo un tipo de entidad:

- **Album:** contiene la información relativa a la posición en la lista, año de salida, nombre del álbum, artista y géneros musicales

Los datos están disponibles en formato CSV. En la siguiente figura se muestran las primeras líneas del fichero de datos.

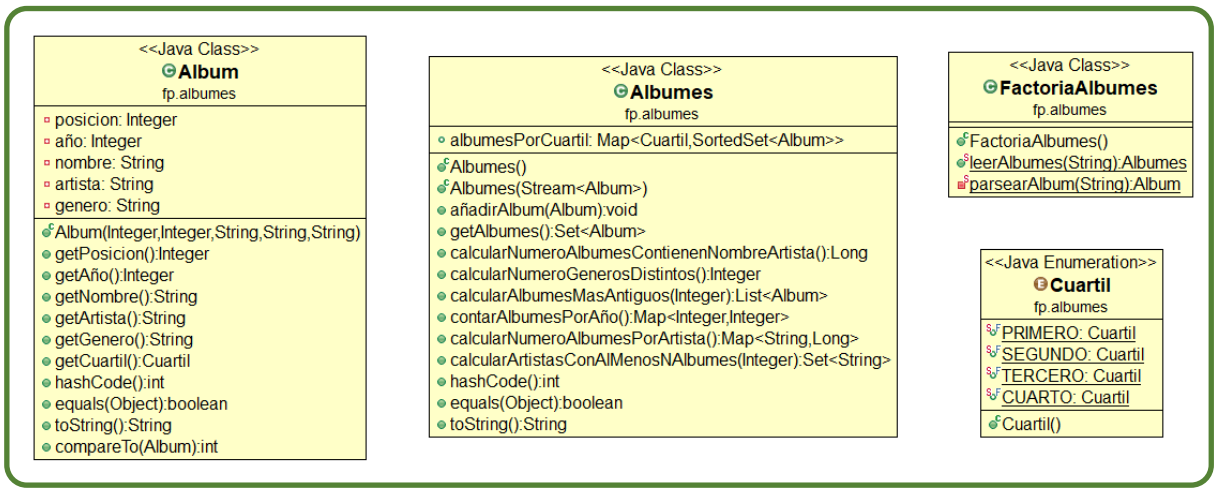
Datos de álbumes

Posición	Año	Álbum	Artista	Género
1	1967	Sgt. Pepper's Lonely Hearts Club Band	The Beatles	Rock
2	1966	Pet Sounds	The Beach Boys	Rock
3	1966	Revolver	The Beatles	Rock
4	1965	Highway 61 Revisited	Bob Dylan	Rock
5	1965	Rubber Soul	The Beatles	Rock/Pop
6	1971	What's Going On	Marvin Gaye	Funk/Soul
7	1972	Exile on Main St.	The Rolling Stones	Rock
8	1979	London Calling	The Clash	Rock
9	1966	Blonde on Blonde	Bob Dylan	Rock/Blues
10	1968	The Beatles ('The White Album')	The Beatles	Rock

3 Modelo

En el siguiente diagrama se muestran todos los elementos que habrá que implementar en esta prueba. Todos ellos se incluirán en el paquete **fp.albumes**. Los aspectos más destacables del modelo son:

- **Cuartil:** tipo enumerado con los cuatro cuartiles en los que distribuiremos los álbumes. Se proporciona **resuelto** junto con el enunciado.
- **Album**: clase para implementar el tipo básico.
- **Albumes**: tipo contenedor que incluye, además, algunos métodos de consulta basados en tratamientos secuenciales.
- **FactoriaAlbumes**: clase para dar soporte a la creación de objetos **Album** y **Albumes** a partir de datos en un fichero CSV. Se proporciona **parcialmente resuelta** junto con el enunciado.



Este diagrama ha sido generado con el plugin de Eclipse ObjectAid
URL de instalación: http://www.objectaid.com/update/current

4 Ejercicios

Para cada ejercicio se muestra la puntuación global. No se puntuarán aquellos fragmentos de código que puedan ser generados de forma automática con Eclipse.

EJERCICIO 1 -- [1.5 puntos]

Crear la clase **Album** con los siguientes atributos y métodos, comprobando las restricciones en los casos en los que sea necesario

- **posicion:** atributo *Integer* con la posición del álbum en la lista (debe estar entre 1 y 500)
- **año:** atributo *Integer* con el año de salida (debe ser mayor que cero)
- **nombre:** atributo *String* con el nombre
- **artista:** atributo *String* con el nombre del artista o agrupación
- **genero:** atributo *String* con el género del álbum
- La clase deber ser **Comparable**
- **Album:** constructor de la clase a partir de los atributos en el orden que se indica en el

ejercicio anterior

- Métodos *getters*: para todos los atributos de la clase
- **Album::getCuartil:** propiedad derivada que se calcula de la siguiente forma. Si la posición está entre 1 y 125 estará en el primer cuartil, si está entre 126 y 250 estará en el segundo, si está entre 251 y 375 estará en el tercero, y a partir de 376 estará en el cuarto. Los cuartiles se representan mediante el tipo enumerado **Cuartil**
- **Album::toString:** mostrando la **posicion**, **nombre** y **artista**
- **Album::equals:** usando **posicion**, **nombre** y **artista**
- **Album::hashCode:** usando la misma selección de atributos que el método **equals**
- **Album::compareTo:** el criterio se establece por **artista**, **nombre** y **posición** (en ese orden)

EJERCICIO 2 -- [7.75 puntos]
Crear la clase **Albumes** con los siguientes atributos y métodos

- **albumesPorCuartil:** atributo de tipo **Map<Cuartil,SortedSet<Album>>**. Al ser un *map* de *set*, en este caso se complican un poco más el método *añadir* y el constructor de *Stream*.

- **Albumes:** constructor sin parámetros
- **Albumes:** constructor a partir de un *Stream* de **Album**, debe crear el diccionario **albumesPorCuartil** a partir de un flujo de **Album**
- **Albumes::añadirAlbum:** método para añadir un **Album** al diccionario **albumesPorCuartil**
- **Albumes:: getAlbumes:** propiedad derivada que devuelve un conjunto con todos los álbumes del diccionario
- **Albumes::toString:** mostrando todos los atributos

- **Albumes::calcularNumeroAlbumesContienenNombreArtista:** devuelve el número de álbumes que contienen en su título el nombre del artista
- **Albumes::calcularNumeroGenerosDistintos:** devuelve el número de géneros que hay en la colección de álbumes
- **Albumes::calcularAlbumesMasAntiguos:** recibe un entero **n** y devuelve una lista con los **n** álbumes más antiguos
- **Albumes::agruparNumAlbumesPorAño**: devuelve un diccionario con el número de álbumes por año
- **Albumes:: calcularArtistasConAlMenosNAlbumes**: recibe un entero **n** y devuelve el conjunto de artistas que tienen al menos **n** álbumes. Se puede usar el método de apoyo **calcularNumeroAlbumesPorArtista** para calcular el número de álbumes de cada artista.

EJERCICIO 3 -- [0.75 puntos]
Crear el siguiente método estático de la clase **FactoriaAlbumes**

- **FactoriaAlbumes::parsearAlbum:** método privado para construir un objeto **Album** a partir de una línea CSV del fichero de entrada