Objetivo general del taller

El objetivo general de este taller es practicar el diseño e implementación de excepciones, utilizándolas para complementar un programa existente.

Objetivos específicos del taller

Durante el desarrollo de este taller se buscará el desarrollo de las siguientes habilidades:

1. Diseñar jerarquías de excepciones a partir de la información sobre un dominio y los posibles problemas que pueden presentarse ahí.
2. Diseñar el contenido de las excepciones (payload) para que sea útil para el diagnóstico o corrección de los problemas.
3. Implementar las excepciones y los mecanismos para su manejo. Implementar un conjunto de pruebas unitarias a partir de su diseño, utilizando un framework de pruebas especializado.
4. Implementar un conjunto de pruebas unitarias a partir de su diseño, utilizando un framework de pruebas especializado.

Instrucciones generales

1. Descargue de Brightspace el archivo `Taller6-Libreria_esqueleto.zip` y descomprímalo en el workspace (carpeta de trabajo) de Eclipse.
2. Importe el proyecto a Eclipse.
3. Realice las modificaciones necesarias al proyecto de acuerdo a los requerimientos que se muestran más adelante. Para manejar los errores relevantes, use el mecanismo de Excepciones de Java siguiendo las restricciones establecidas en el enunciado.

El taller debe desarrollarse en los grupos de trabajo del semestre.

Descripción del caso: Librería

Para este taller vamos a trabajar sobre una aplicación que maneja la información de una Librería.

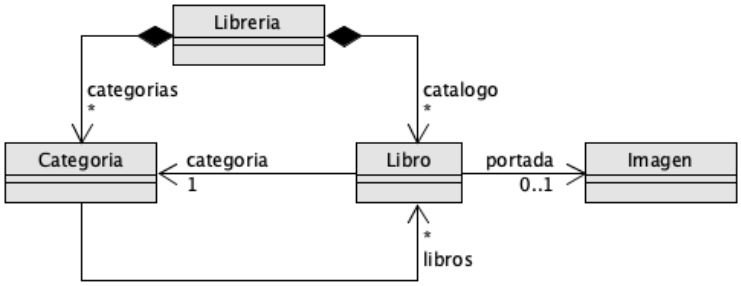
La información que usaremos dentro de la librería es un subconjunto del *dataset* disponible en el siguiente url:

https://www.kaggle.com/lukaanicin/book-covers-dataset

Este *dataset* incluye tanto la información de los libros como las portadas.

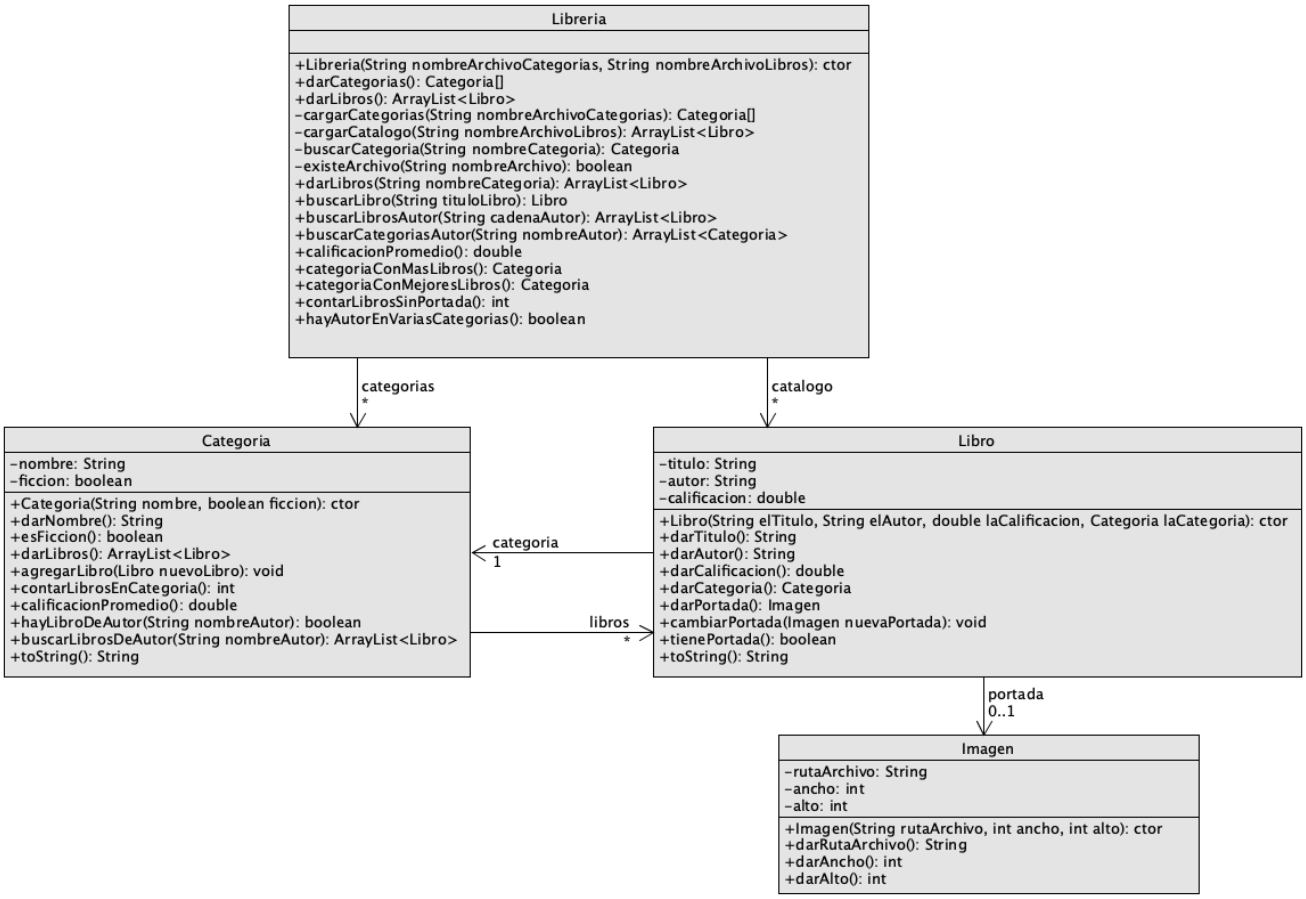
Clases involucradas

El corazón de este taller son 4 clases que pueden verse en el siguiente diagrama de clases resumido.



- La clase `Imagen` sirve para encapsular la información de una imagen, incluyendo el nombre del archivo y sus dimensiones.
- La clase `Libro` representa a un libro dentro de la librería y mantiene su información básica. Además, los libros pueden o no tener una portada, la cual se representa con una instancia de `Imagen`.
- La clase `Categoría` sirve para agrupar libros dentro de la librería. Cada categoría tiene un nombre y se sabe si es una categoría de ficción o no. Además, cada categoría sabe qué libros pertenecen a ella. Podría darse el caso de que una categoría no tenga libros en un momento dado.
- La clase `Librería` es la clase dentro de este modelo que tiene más responsabilidades funcionales: una librería tiene un conjunto de categorías y tiene un catálogo de libros, pero además es capaz de hacer búsquedas sobre los libros y sobre las categorías.

El siguiente diagrama muestra de forma un poco más detallada las clases del modelo que acabamos de describir:



Parte 1: Libros en categorías que no existen

Modifique el archivo de libros para que algunos libros pertenezcan a categorías que no existen en el archivo de categorías.

Modifique la aplicación para que cuando se cargue el archivo de libros se creen las categorías faltantes y se le avise al usuario cuáles fueron las categorías nuevas que fueron creadas y cuántos libros tiene cada una. Por ejemplo, si en el archivo de libros hay dos marcados en la categoría “Ficcción” y uno en la categoría “Sciencia”, deberían agregarse las categorías “Ficcción” y “Sciencia”, los tres libros deberían agregarse dentro de la categoría correspondiente, y al usuario se le debería advertir de las nuevas categorías que fueron creadas.

Parte 2: Renombrar categorías

Agregue a la aplicación la funcionalidad para renombrar una categoría, pero manteniendo la restricción de que no puede haber dos categorías con el mismo nombre. El método que agregue a la clase Librería **debe** lanzar una excepción si el nombre de la categoría está repetido y la interfaz debe encargarse de informarle al usuario sobre el error.

Restricción: La interfaz no puede ser la encargada de revisar si el nombre de una categoría ya existe.

Parte 3: Borrar libros

Agregue a la aplicación la funcionalidad para eliminar libros. Cuando el usuario seleccione esta opción, el programa debe permitirle digitar uno o varios nombres de autores, separados por comas (en un solo campo de texto tiene que escribir todos los nombres). La aplicación debe intentar borrar todos los libros de esos autores, pero sólo debe realizar la operación de borrado si para todos los autores que digitó el usuario existía al menos un libro en la librería. Si la operación se pudo realizar, al usuario se le debe informar la cantidad de libros borrados. Si la operación no se pudo realizar, al usuario se le debe informar cuáles de los nombres digitados correspondían a autores y cuáles no. Dicho de otra forma, no puede pasar que al terminar de ejecutar la operación se hayan borrado libros si dentro de la lista de autores al menos uno no existía.

El método para eliminar libros de la clase Librería debe generar una excepción si los libros no podían eliminarse. Esa excepción debe tener toda la información de los libros que no pudieron eliminarse que se le tiene que informar al usuario.

Restricción: La interfaz no puede ser la encargada de revisar si cada uno de los autores existe o no.

Parte 4: Pruebas unitarias

En esta parte del taller, ustedes tendrán que diseñar e implementar pruebas unitarias y de integración utilizando el framework JUnit. Idealmente se deberían implementar primero pruebas unitarias pero el proyecto sobre el que ustedes trabajarán tiene pocas clases, pero muy acopladas entre ellas, lo cual hace difícil probarlas por separado.

Las pruebas que ustedes van a construir deben procurar que todos los métodos de las clases sean verificados, una parte importante del esfuerzo que tendrán que hacer corresponde al diseño de los escenarios de prueba.

En esta parte deben completar las clases *ImagenTest* y *LibroTest*. Primero comiencen completando los métodos *setUp* y *tearDown* para cada clase, estos van a ser usados en todas las pruebas y son fundamentales para que las pruebas funcionen correctamente. Luego de completar los métodos mencionados anteriormente, deben completar todos los métodos de las clases. Para esto, deben reemplazar los *fail* por los vistos en clase.

Entrega

1. Cree un repositorio privado para la organización donde quedará el taller 6 y dejen ahí todos los archivos relacionados con su entrega.

2. Entregue a través de Brightspace el URL para el repositorio, en la actividad designada como "**Taller 6**". Sólo se debe hacer una entrega por grupo.