Objetivo general del taller

El objetivo de este taller es aprender a implementar en Java un modelo de clases sencillo, reconocer los diferentes componentes que podrían aparecer en un modelo de clases y practicar la implementación de algoritmos básicos en Java.

Objetivos específicos del taller

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

1. Implementar en Java un modelo de clases expresado como un diagrama de clases de UML, incluyendo atributos de los principales tipos básicos y métodos con funcionalidades básicas.
2. Entender el significado de la cardinalidad (*multiplicity*) en las asociaciones entre clases y ser capaz de implementar diferentes tipos de asociaciones usando relaciones directas, arreglos y listas (`ArrayList`).
3. Entender el significado de `null` en Java, saber verificar si una referencia tiene valor `null` y entender los problemas relacionados con invocar métodos sobre referencias con valor `null`.
4. Recorrer arreglos y listas utilizando `while`, `for` y `for-each`.
5. Conocer el método `equals`.
6. Utilizar con habilidad el ambiente de programación del curso y ser capaz de buscar recursos externos que les permitan resolver problemas técnicos puntuales.

Instrucciones generales

1. Descargue de Brightspace el archivo `Taller1-Libreria_esqueleto.zip` y descomprímalo en el workspace (carpeta de trabajo) de Eclipse.
2. Importe el proyecto a Eclipse.
3. Lea con cuidado este documento y vaya realizando las actividades una por una. En el esqueleto encontrará marcados con la etiqueta **`TODO`** (to-do, por hacer) los puntos en los cuales usted debe completar algo. Lea el documento al tiempo que vaya resolviendo el taller: en este documento encontrará información importante para completar el programa entendiendo lo que está haciendo.

El taller puede desarrollarse en grupos de 2 o 3 estudiantes: todos los miembros del grupo deben participar en la realización del taller y son responsables por el resultado.

No solo es posible usar recursos externos (Google, StackOverflow, libros, etc.) para resolver el taller, sino que es deseable que se consulten fuentes externas.

Descripción del caso: Librería

Para este taller vamos a trabajar sobre una aplicación que maneja la información de una Librería. En el siguiente enlace pueden encontrar un video donde se ve la aplicación funcionando después de completar todas las actividades del taller:

https://youtu.be/bi00yazla_A

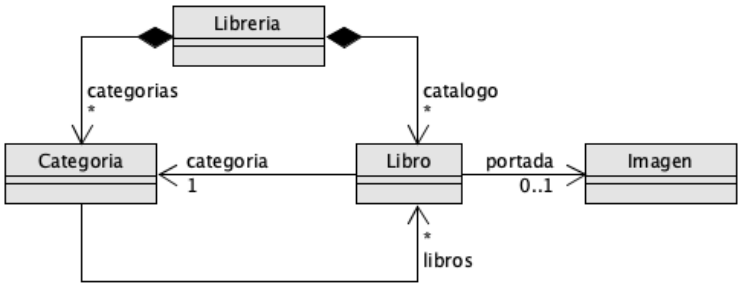
La información que usaremos dentro de la librería es un subconjunto del *dataset* disponible en el siguiente url:

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

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

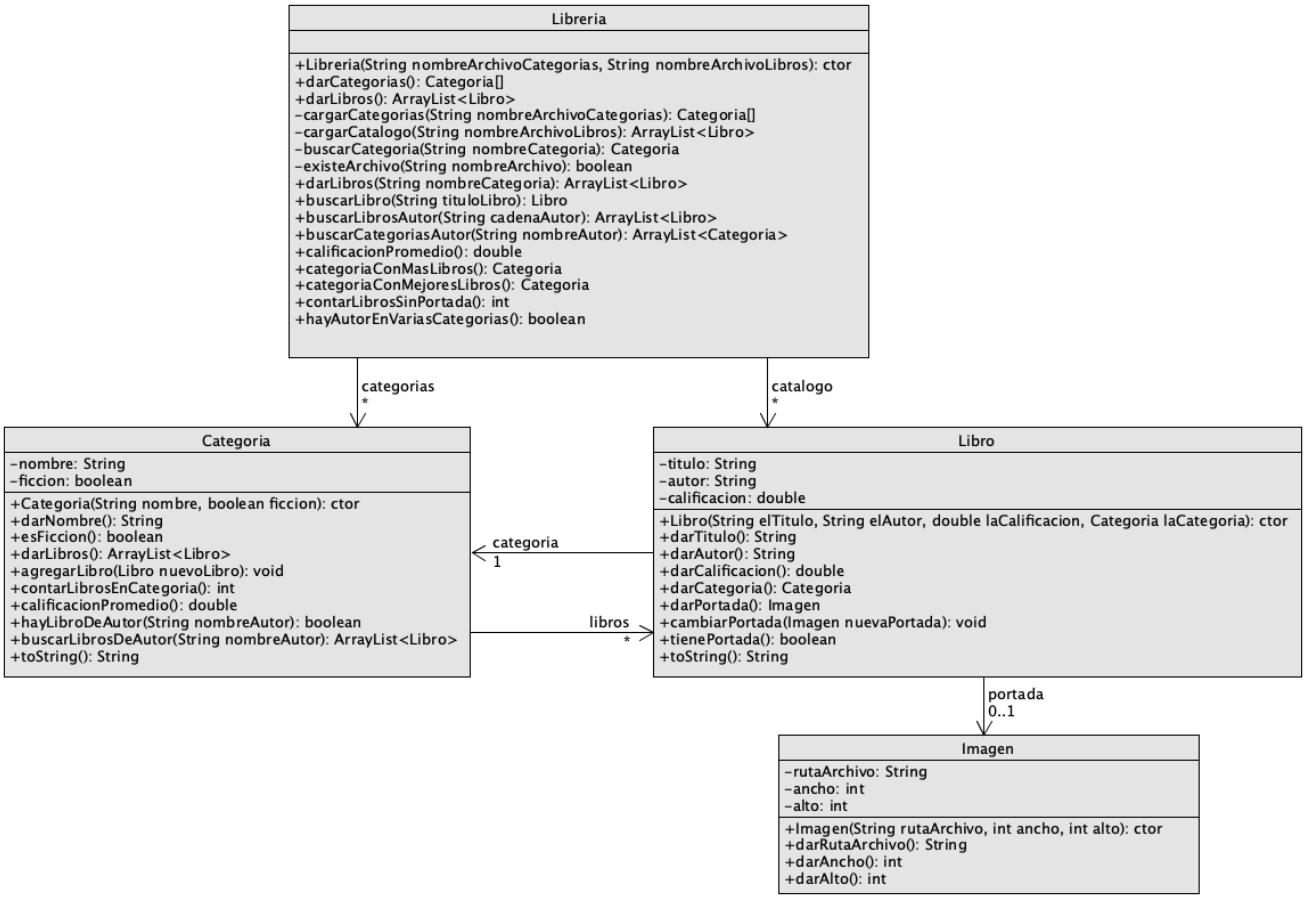
Clases involucradas

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 0: Revisar las clases del modelo

Revise el código de las clases `Librería`, `Libro`, `Categoria` e `Imagen`. Estas clases tienen responsabilidades muy precisas: además de manejar su propio estado, todas estas clases exponen métodos públicos para resolver preguntas particulares.

Para resolver los ejercicios de este taller, le será de **MUCHA UTILIDAD** saber qué métodos ofrece cada clase. De esta forma no tendrá que volver a implementar cosas que ya estén implementadas en otras clases. Además, re-implementar algunas de estas cosas lo obligarían a romper el encapsulamiento y esto no es recomendable.

Dentro del código encontrará la definición de atributos, retornos, parámetros y variables con tipos que usted tal vez no conozca todavía. Al final del taller debería reconocerlos todos, pero acá vamos a introducir los que tienen la sintaxis más compleja.

Teoría: Arreglos y Listas en Java

En Java, los arreglos son estructuras de datos lineales donde todos los datos deben ser del mismo tipo. Se expresan usando los caracteres `[]` después del tipo contenido en el arreglo. Es decir que un arreglo de cadenas (`String`) se expresa usando:

```
String[]
```

Por otro lado, las listas pueden expresarse de muchas maneras diferentes dependiendo de cómo estén implementadas. En uno de los próximos talleres estudiaremos más sobre estas posibilidades, así que en este taller vamos a usar sólo las listas implementadas sobre arreglos que encontramos en la clase `ArrayList`. Además, en Java las listas aprovechan un mecanismo llamado *generics* (también lo estudiaremos más adelante), el cual nos permite fijar el tipo de cosas que vamos a meter dentro de una lista. De esta forma, el tipo de una lista de cadenas va a ser diferente del tipo de una lista de libros, así que el compilador nos ayuda a evitar errores donde metamos elementos del tipo equivocado en una lista. Los tipos de estas listas de ejemplo serían los siguientes:

```
ArrayList<String>
ArrayList<Libro>
```

Finalmente, tenga en cuenta que para poder usar una lista dentro de una clase tiene que haber importado ArrayList dentro de su clase, usando la siguiente instrucción:

```
import java.util.ArrayList;
```

Teoría: Comparar valores en Java

En Java, hay dos formas de comparar valores. Cuando se trata de **tipos simple**s (`int`, `float`, `boolean`, `long`, etc.) se debe usar el operador `==`. Es decir que si tenemos una variable llamada `valor` y queremos saber si tiene el valor 5 podemos usar la siguiente expresión:

```
valor == 5
```

Por otro lado, para comparar objetos NO SE DEBE USAR `==`. Si usamos el operador `==` cuando comparemos dos objetos el resultado será similar al que habríamos obtenido con el operador `is` en Python. Es decir, cuando en Java comparamos dos objetos usando == no estamos preguntando si son iguales, estamos preguntando si son el mismo. Para comparar si dos objetos son iguales, debe usarse el método `equals`. Veamos ahora unos ejemplos de esto usando la clase String[1]:

```
String cadena1 = “Hola”;
String cadena2 = “Mundo”;
String cadena3 = cadena1 + cadena2;
String cadena4 = “HolaMundo”;
boolean b1 = (cadena1 == cadena2);                // false
boolean b2 = (cadena1 == cadena1);                // true
boolean b3 = (cadena3 == cadena1+cadena2);        // false
boolean b4 = (cadena3 == cadena4);                // false
boolean b5 = (cadena4.equals(cadena1+cadena2));   // true
boolean b6 = (cadena3.equals(cadena4));           // true
```

Finalmente, tenga en cuenta que para que tenga sentido usar el método `equals` dentro de una clase que usted haya definido, debería haber *sobrecargado* el método. Eso lo estudiaremos en una próxima clase.

[1] La clase `String` es la más fácil de usar para dar ejemplos, pero también tiene algunas particularidades ligadas a optimizaciones que hace la JVM para que manejar cadenas sea mucho más rápido. En particular, si en su código aparecen explícitamente dos cadenas idénticas, la optimización las convertirá en una sola y su código podría comportarse de formas inesperadas. No es necesario que conozca ahora todos los detalles: sólo asegúrese de siempre usar `equals` para comparar objetos y no debería tener sorpresas.

Parte 1: Asociación Libro – Imagen (portada)

La multiplicidad de la asociación `'portada'` entre las clases `Libro` e `Imagen` es 0..1 en el diagrama de clases, lo cual indica que un libro podría tener, o no, una imagen de portada. En esta parte del taller usted va a completar la clase `Libro` para establecer esta asociación, teniendo cuidado de manejar los casos en que un libro NO tenga una imagen asociada y el atributo de la clase tenga el valor `null`.

Actividades y ayuda

Complete todos los lugares marcados como `TODO Parte 1` dentro de la clase `Libro`.

Para saber si un atributo o una variable tiene el valor nulo, sólo necesita compararlo con `null`:

```
if (variable == null)
{
    // variable tiene el valor nulo
}
```

Parte 2: Asociación Libreria – Categoria (categorias)

Entre las clases `Librería` y `Categoría` existe una asociación llamada `categorías` que vamos a representar como un arreglo de objetos de la clase `Categoría`. En esta parte del taller, usted va a completar la clase `Librería` para representar esta relación y para practicar algorítmica sobre arreglos.

Actividades y ayuda

Complete todos los lugares marcados como `TODO Parte 2` dentro de la clase `Libreria`.

El tipo para un arreglo de objetos de tipo `Categoria` se expresa como: `Categoria[ ]`

Puede recorrer un arreglo usando las estructuras `while`, `for` o `for-each`. Los siguientes bloques muestran cómo recorrer un arreglo llamado `palabras` usando cada una de estas estructuras.

```
int i = 0;
while (i < palabras.length)
{
	System.out.println(palabras[i]);
	i++;
}

for (int j = 0; j < palabras.length; j++)
{
	System.out.println(palabras[j]);
}

for (String palabra : palabras)
{
	System.out.println(palabra);
}
```

Parte 3: Asociación Categoria – Libro (libros)

Entre las clases `Categoria` y `Libro` existe una asociación llamada `libros` que vamos a implementar usando una lista de libros (un `ArrayList` de `Libro`). En esta parte del taller, usted va a completar la clase `Categoria` para representar esta relación, practicará la algorítmica sobre listas y tendrá que consultar y aprender a usar algunos de los métodos que ofrece la clase `ArrayList`.

Actividades y ayuda

Complete todos los lugares marcados como `TODO Parte 3` dentro de la clase `Categoria`.

El tipo para una lista de objetos de tipo `Libro` se expresa como: `ArrayList<Libro>`

Puede recorrer una lista usando las estructuras `while`, `for` o `for-each`. Los siguientes bloques muestran cómo recorrer una lista llamada `palabras` usando cada una de estas estructuras.

```
int i = 0;
while (i < palabras.size())
{
	System.out.println(palabras.get(i));
	i++;
}

for (int j = 0; j < palabras.size(); j++)
{
	System.out.println(palabras.get(j));
}

for (String palabra : palabras)
{
	System.out.println(palabra);
}
```

Parte 4: Asociación Librería – Libro (catalogo)

Entre las clases `Librería` y `Libro` existe una asociación llamada `catalogo` que vamos a representar como una lista objetos de la clase `Libro`. En esta parte del taller, usted va a completar la clase `Librería` para representar esta relación y para seguir practicando algorítmica sobre listas.

Actividades

Complete todos los lugares marcados como `TODO Parte 4` dentro de la clase `Libreria`.

Parte 5: Reflexión y revisión

Resuelva las siguientes preguntas con base en lo que aprendió en el taller y deje sus respuestas en un archivo dentro de la carpeta `docs` del proyecto.

Tipos básicos de Java

1. ¿Cuáles tipos básicos de Java se utilizaron en el taller y qué se puede representar con cada uno?

null

2. ¿A qué elemento de Python se parece el valor `null` de Java?

3. ¿Qué diferencias hay entre el valor `null` de Java y el valor que respondió para la pregunta anterior?

Arreglos y listas

4. ¿Cómo puede averiguar el tamaño de un arreglo?

5. ¿Cómo puede averiguar el tamaño de una lista?

6. ¿Cuál es la principal restricción de un arreglo con respecto a las listas?

7. ¿Cómo se especifica el tipo de una lista de números enteros?

8. Haga una lista de los métodos de los arreglos que le hayan sido de utilidad para este taller. ¿Algún comentario?

9. Haga una lista de los métodos de las listas que le hayan sido de utilidad para este taller. ¿Algún comentario?

Instrucciones iterativas en Java

10. ¿Qué diferencias hay entre las tres estructuras para construir instrucciones iterativas? (`while`, `for` y `for-each`)

Entrega

1. Comprima la carpeta de trabajo y entréguela como un archivo .zip con nombre '`taller1_login1_login2_login3.zip`'. El archivo .zip debe incluir el proyecto de Eclipse con todos sus componentes – excepto la carpeta '`data`'. El archivo .zip también debe incluir un documento donde se respondan las preguntas de la última parte del taller, ubicado dentro de la carpeta `docs`.

Por favor, **NO INCLUYA LA CARPETA DATA**: se le reducirán puntos en la calificación si incluye esta carpeta.

2. Entregue el archivo comprimido a través de Brightspace en la actividad designada como "**Taller 1**". Sólo se debe hacer una entrega por grupo.