Programación Modular y Orientación a Objetos - Laboratorio 5

Requisitos previos

El alumnado se desenvuelve correctamente en el entorno Eclipse y maneja adecuadamente la perspectiva Java para resolver ejercicios que manejan varias clases a la vez, incluyendo MAEs y TADs. Además, es capaz de crear y representar mediante diagramas UML clases con atributos y métodos sencillos, así como relaciones simples entre clases. Por último, sabe realizar casos y suites de prueba para estas clases con el framework JUnit.

Objetivos

Este laboratorio sirve para afianzar los conocimientos adquiridos en laboratorios anteriores, y su objetivo consiste en reforzar todos los conceptos tratados anteriormente.

Al finalizar este laboratorio, el alumnado deberá ser capaz de:

- Realizar con soltura ejercicios que incluyan clases que implementen tanto TADs como MAEs.
- Dominar el uso de listas (*ArrayList*) e iteradores.
- Efectuar con facilidad pruebas con JUnit.

Motivación

Este laboratorio vuelve a presentar un único ejercicio de cierta complejidad, que en este caso incluye 5 clases interrelacionadas. El diseño de la solución a implementar viene dado en el enunciado, de manera que el laboratorio se centrará exclusivamente en la implementación.

Tarea única: gestión de la Biblioteca Municipal

En esta actividad se va a implementar la gestión de la **Biblioteca Municipal del Barrio**, en lo referente a préstamos de libros a los usuarios. La figura siguiente recoge el diagrama UML con las clases que se van a implementar.

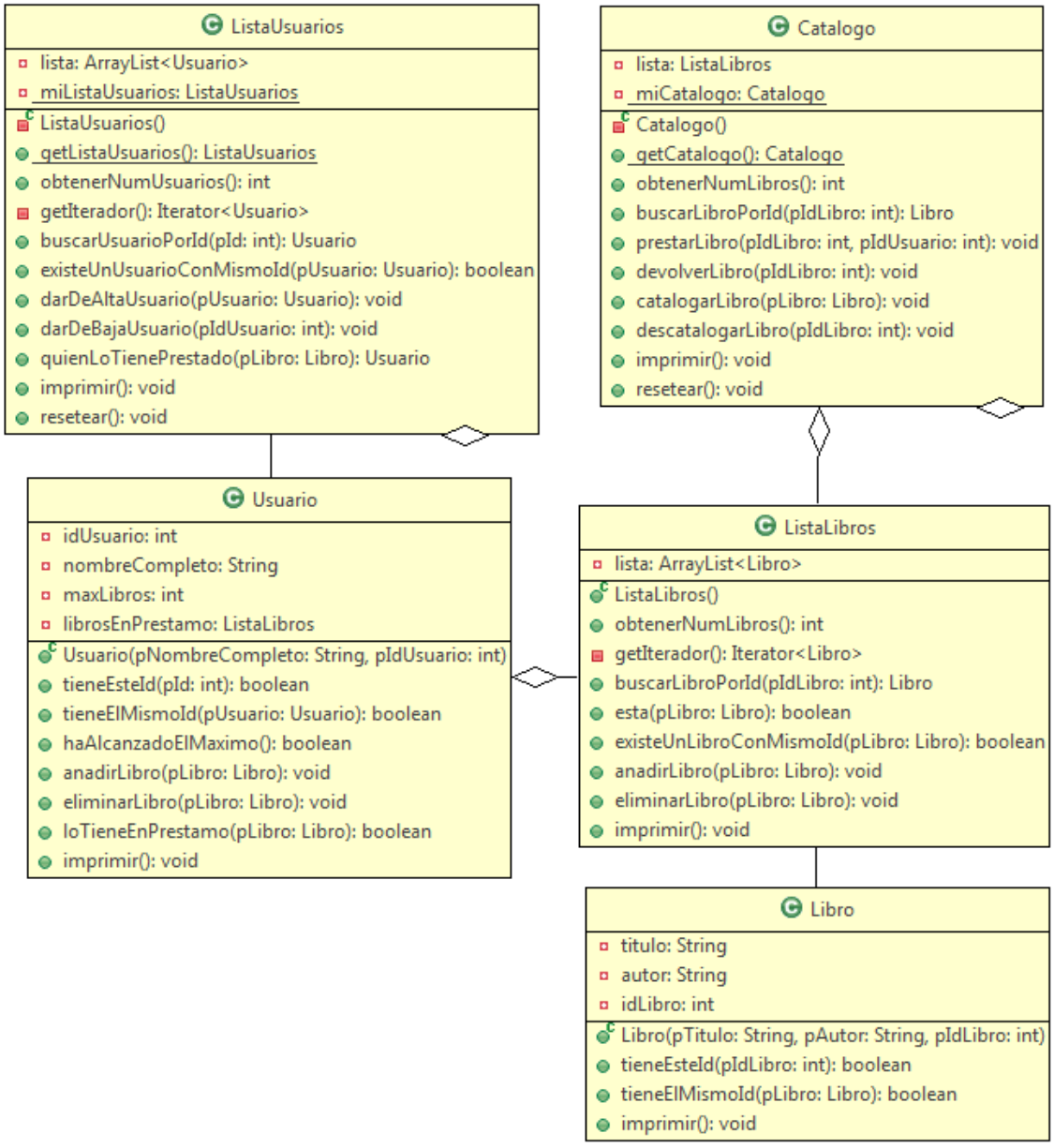

Figura 1 - Diagrama de clases para la gestión de la Biblioteca Municipal del Barrio.

La aplicación debe guardar la información relativa a la relación de usuarios de la biblioteca. Por cada usuario se dispone de un identificador, su nombre completo, el número máximo de libros que puede tomar en préstamo y la relación de libros que tiene prestados.

También debe almacenarse la información del catálogo de la biblioteca, esto es, la lista de libros que hay en la misma. Por cada libro se dispone de un identificador, su título y el nombre del autor.

Las operaciones básicas que se desea implementar incluyen el registro de nuevos usuarios y libros, la búsqueda de libros o usuarios a partir de su identificador, el préstamo de libros a los usuarios, la devolución de un libro que está prestado, y la escritura en la consola del sistema de la información relativa a los libros del catálogo o a los usuarios.

Se pide implementar las clases siguientes, cuyos atributos y métodos se describen a continuación, así como las pruebas unitarias que permitan verificar su corrección.

- **Clase Libro**
  - **Atributos:** el título (String), el autor (String) y un identificador (int).
  - **Método constructor:** recibe como parámetro los valores con los que inicializar los atributos.
  - **Otros métodos:**
    - *tieneEsteId()* devuelve un booleano que indica si el identificador del libro es igual al identificador que recibe como parámetro.
    - *tieneElMismoId()* devuelve un booleano que indica si el identificador del libro es igual al identificador del libro que recibe como parámetro. Los métodos *tieneEsteId()* y *tieneElMismoId()* realizan la misma función, pero se diferencian en que el primero recibe un identificador como parámetro y el segundo recibe un libro.
    - *imprimir()* escribe un mensaje en la consola del sistema mostrando los datos del libro. El método *testImprimir()* de la clase LibroTest da una idea de cómo podría mostrarse dicho mensaje.

- **Clase ListaLibros**
  - **Atributo:** una lista de libros, implementada en este caso como ArrayList.
  - **Método constructor:** crea una instancia vacía de ArrayList<Libro> y se la asigna al atributo lista.
  - **Otros métodos:**
    - *obtenerNumLibros()* devuelve el número de elementos que contiene la lista de libros.
    - *getIterador()* devuelve un elemento de tipo Iterator para que los métodos de la clase ListaLibros puedan recorrer la lista con él. Como siempre, este método será privado.
    - *buscarLibroPorId()* devuelve el libro que tiene como identificador el valor que recibe como parámetro. Si no existe tal libro, devolverá el valor null.
    - *esta()* devuelve un booleano que indica si la lista contiene el libro que recibe como parámetro.

- *existeUnLibroConMismoId* devuelve un booleano que indica si la lista contiene un libro con el mismo identificador que el del libro que recibe como parámetro.
- *anadirLibro()* añade a la lista el libro que recibe como parámetro, excepto si dicho libro ya se encuentra en la lista, en cuyo caso no lo vuelve a añadir.
- *eliminarLibro()* elimina de la lista el libro que recibe como parámetro.
- *imprimir()* escribe la información relativa a cada uno de los libros de la lista. El método *testImprimir()* de la clase ListaLibrosTest da una idea de cómo podría mostrarse dicha información.

- **Clase Usuario**
  - **Atributos:** un identificador (int), el nombre completo (String), el número máximo de libros que puede tomar prestados (int) y la lista de libros que tiene en préstamo (ListaLibros).
  - **Método constructor:** recibe como parámetro los valores con los que inicializar los atributos nombreCompleto e idUsuario. Inicializa maxLibros con el valor por defecto, que es 3, y crea y asigna al atributo librosEnPrestamo una lista de libros vacía.
  - **Otros métodos:**
    - *tieneEsteId()* devuelve un booleano que indica si el identificador del usuario es igual al identificador que recibe como parámetro.
    - *tieneElMismoId()* devuelve un booleano que indica si el identificador del usuario es igual al identificador del usuario que recibe como parámetro. Los métodos *tieneEsteId()* y *tieneElMismoId()* realizan la misma función, pero se diferencian en que el primero recibe un identificador como parámetro y el segundo recibe un usuario.
    - *haAlcanzadoElMaximo()* devuelve un booleano que indica si el usuario tiene en préstamo el máximo de libros que puede tomar prestados.
    - *anadirLibro()* recibe un objeto de tipo Libro y lo añade a la lista de libros prestados que tiene el usuario.
    - *eliminarLibro()* recibe un objeto de tipo Libro y lo elimina de la lista de libros que tiene en préstamo el usuario.
    - *loTieneEnPréstamo()* devuelve un booleano que indica si el usuario tiene el libro que recibe como parámetro en su lista de libros prestados.
    - *imprimir()* escribe un mensaje con los datos del usuario, indicando además si tiene libros en préstamo o no. En caso de tener al menos un libro prestado, escribirá además la información de todos los libros que tiene en su lista. El método *testImprimir()* de la clase UsuarioTest ofrece ejemplos para ambos casos.

- **Clase Catalogo (MAE)**
  - **Atributos:** una lista de libros (ListaLibros) y la instancia (única) de Catalogo.
  - **Método constructor:** inicializa el atributo de tipo lista de la clase.
  - **Otros métodos:**
    - *getCatalogo()* devuelve la única instancia de Catalogo para que pueda ser visible desde cualquier otra clase.
    - *obtenerNumLibros()* devuelve el número de libros que tiene la lista de libros del catálogo.
    - *buscarLibroPorId()* devuelve el libro que tiene como identificador el valor que recibe como parámetro. Si no existe tal libro, devolverá el valor null.
    - *prestarLibro()* recibe como parámetro los identificadores de un libro y de un usuario, y añade el libro a la lista de libros del usuario, salvo que el usuario tenga prestados el máximo de libros permitidos o el libro ya esté prestado a otro usuario, en cuyo caso se mostrará un mensaje por la consola del sistema avisando de la circunstancia correspondiente.
    - *devolverLibro(),* tras averiguar qué usuario lo tiene prestado, elimina de su lista de libros prestados el libro cuyo identificador recibe como parámetro.
    - *catalogarLibro()* añade al catálogo el libro que recibe como parámetro, salvo que ya exista otro libro con el mismo identificador, en cuyo caso el libro no se añadirá y se mostrará un mensaje por la consola del sistema.
    - *descatalogarLibro()* elimina del catálogo el libro cuyo identificador recibe como parámetro, excepto que esté prestado a un usuario, en cuyo caso no se eliminará sino que se mostrará un mensaje indicando tal circunstancia.
    - *imprimir()* escribe un mensaje indicando cuántos libros tiene el catálogo, así como la información relativa a cada uno de ellos. El método *testImprimir()* de la clase CatalogoTest da una idea de cómo podría mostrarse dicha información.
    - *resetear()* vacía la lista de libros del catálogo. La utilidad de este método vendrá a la hora de realizar las pruebas unitarias, ya que en principio no debería existir en una aplicación real, y menos si su visibilidad es pública.

- **Clase ListaUsuarios (MAE)**
  - **Atributos:** una lista de usuarios, implementada en este caso como ArrayList, y la instancia (única) de ListaUsuarios.
  - **Método constructor:** inicializa el atributo de tipo lista de la clase.

- **Otros métodos:**
    - *getListaUsuarios()* devuelve la única instancia de ListaUsuarios para que pueda ser visible desde cualquier otra clase.
    - *obtenerNumUsuarios()* devuelve el número de usuarios que contiene la lista.
    - *getIterador()* devuelve un elemento de tipo Iterator para que los métodos de la clase ListaUsuarios puedan recorrer la lista con él. Como siempre, este método será privado.
    - *buscarUsuarioPorId()* devuelve el usuario que tiene como identificador el valor que recibe como parámetro. Si no existe tal usuario, devolverá el valor null.
    - *existeUnUsuarioConMismoId()* devuelve un booleano que indica si la lista contiene un usuario cuyo identificador sea igual al del usuario que recibe como parámetro.
    - *darDeAltaUsuario()* añade a la lista el usuario que recibe como parámetro, salvo que ya exista un usuario con el mismo identificador, en cuyo caso el nuevo usuario no se añadirá y se mostrará un mensaje por la consola del sistema que avise de tal circunstancia.
    - *darDeBajaUsuario()* elimina de la lista de usuarios el usuario cuyo identificador recibe como parámetro.
    - *quienLoTienePrestado()* recibe un libro como parámetro y devuelve el usuario (de la lista de usuarios) que tiene dicho libro en su lista de libros prestados. Si no existe tal usuario, devuelve el valor null.
    - *imprimir()* escribe un mensaje indicando cuántos usuarios hay en la lista de usuarios, y a continuación escribe la información relativa a cada uno de ellos. El método *testImprimir()* de la clase ListaUsuariosTest da una idea de cómo podría mostrarse dicha información.
    - *resetear()* vacía la lista de usuarios. La utilidad de este método vendrá a la hora de realizar las pruebas unitarias, ya que en principio no debería existir en una aplicación real, y menos si su visibilidad es pública.