PRÀCTICA 3: LIBRARY MANAGER

Informe

Autor: Èric Bitrià Ribes
DNI: 49755704B
Grup: 1

Índex

1. Introducció 2
1.1 Explicació de la pràctica 2
2. La funció addBook 3
2.1 La funció auxiliar BooksFile.length() 3
3. La funció lendBook 4
3.1 La funció bookExists i memberExists 4
3.2 La funció isLent 4
3.3 La funció canBorrow 5
3.3.1 La funció countBooks 5
4. La funció returnBook 6
4.1 La funció containsBook 6
5. Conclusions 7

1. Introducció

En aquest informe es fa una explicació general del funcionament de la pràctica 3: “LibraryManager”, amb els objectius que pretén assolir, que en aquest cas serien la manipulació de fitxers a baix nivell com seria en formats binaris i altres formats de text.

A continuació, després de l’explicació inicial de pràctica, es mostren els tres mètodes més complicats d’implementar així com les seves funcions auxiliars. Cada mètode requerirà d’un raonament explicat en forma de diagrama per facilitar la seva comprensió.

Finalment s’exposaran unes breus conclusions i valoracions generals de la pràctica.

1.1 Explicació de la pràctica

LibraryManager es tracta d’un programa que ens permet organitzar els llibres i membres d’una llibreria sota un sistema d’ID. Cada membre disposarà dels paràmetres següents:

- **Nom:** Nom del membre
- **Adreça:** Lloc de residència
- **ID:** Identificador que ens permetrà operar amb el membre.

Cada membre disposarà d’un màxim de llibres que podrà demanar, definit en la classe “Member” al valor MAX_BOOKS.

D’altra banda cada llibre disposarà de dos paràmetres:

- **Títol:** Títol del llibre
- **ID:** Identificador del llibre per operar amb ell internament.

Amb aquests elements el libraryManager tindrà que executar diferents comandes:

- **ADDBOOK:** Afegir un nou llibre.
- **ADDMEMBER:** Crear un nou soci.
- **LEND:** Deixar un llibre a un soci.
- **RETURN:** Retornar un llibre associat a un soci.

Totes les operacions per executar seran emmagatzemades en un fitxer anomenat “moviments.csv” i tindrem un fitxer anomenat “manager.log” que s’encarregarà de guardar els missatges d’error o confirmació de totes les operacions realitzades.

Seguidament, tota la informació relacionada amb els membres i els llibres serà emmagatzemada i modificada en format binari en els arxius “booksFile” i “memberFile”.

2. La funció addBook

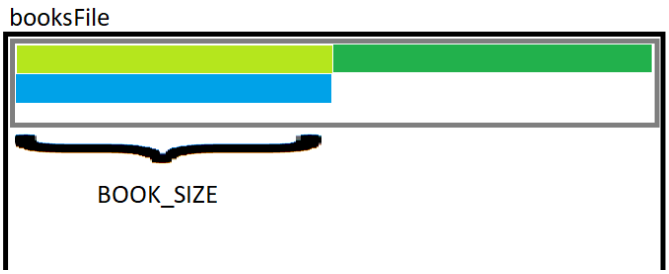
La funció addBook realitza l'acció d'afegir un nou llibre. Rep com a paràmetres el títol del llibre en format string i un int que representarà el número de l'operació que estarà realitzant.

Per afegir cada llibre seran necessaris dos paràmetres el ID del llibre, representat com un nombre en format long, i el títol en format string. En aquest cas ja disposem del títol, paràmetre que hem obtingut al cridar la funció. Solament ens falta calcular el id.

Per calcular el ID del llibre s'aprofitarà la forma en la que s'atribueixen els ID en relació amb la llargada del fitxer que conté tots els llibres.

2.1 La funció auxiliar BooksFile.length()

Aquesta funció auxiliar ens retorna el nombre de llibres continguts dins del fitxer. Per calcular el nombre de llibres utilitza la llargada en bytes del fitxer en relació a la llargada en bytes de cada llibre:



Il·lustració 1: Representació del arxiu booksFile amb diagrames de colors representant la llargada de cada llibre en bytes.

Al saber que tots els llibres tindran exactament la mateixa mida, el fet de dividir la mida en bytes que ocupa cada llibre respecte la mida total del fitxer ens retornarà el nombre de llibres que aquest conté. D'aquesta manera ens assegurem que cada ID serà únic per cada llibre al anar incrementant-se a mesura que n'afegim de nous.

```
public long length() throws IOException {
        return this.raf.length()/Book.SIZE;
    }
```

Una vegada creat el llibre l'hem d'escriure dins el fitxer booksFile amb la comanda write implementada en la classe BooksFile. Seguidament escrivim el missatge de que hem afegit el nou llibre al log.

De forma similar s'implementa la funció addMember, la única diferència és que com a funcionalitat addicional comprovem que el nou membre ja estigui afegit prèviament. També es podria incorporar aquesta funció al addBooks però, al tractar-se d'un gestor de llibreria, es possible que es disposin de diverses copies del mateix llibre, i per tant tenir una comprovació de si el llibre està repetit seria un impediment innecessari.

3. La funció lendBook

La funció lendBook realitza el préstec del llibre a un membre. Rep com a paràmetres el ID del llibre seleccionat i el ID del membre, apart també rep un int amb el nombre de l'operació que s'està realitzant útil per després imprimir tant els missatges d'error com d'èxit.

Abans de realitzar el préstec s'han de fer múltiples comprovacions:

- **Existeix el llibre:** Comprovar que el ID del llibre existeix.
- **Existeix el membre:** Comprovar que el ID del membre existeix.
- **El llibre ja el té un altre membre:** Veure si està ocupat.
- **El membre pot agafar més llibres:** Veure si el membre té el màxim de llibres o no.

Cada comprovació serà successora de l'anterior i per tant l'estructura de la funció lenBook tindrà un format descendent amb les comprovacions corresponents. Per implicar cada comprovació es crearan una sèrie de funcions auxiliars booleanes:

3.1 La funció bookExists i memberExists

La funció bookExists comprova que el ID del llibre enviat com a paràmetre estigui dins del fitxer bookFiles. Tenint en compte que cada ID del llibre es un valor numèric i representa la seva posició dins el fitxer binari (veure il·lustració 1 del apartat 2.1 no hi ha buits entre els llibres, i solament hem de comprovar que el ID es trobi dins del rang de llibres ja afegits:

```
private boolean bookExists(long bookId) throws IOException{
        return bookId < booksFile.length() && bookId>=0;
    }
```

Si es troba dins del rang retornarà "true" i sinó "false".

La funció memberExists funciona exactament de la mateixa manera l'unic que realitza la comprovació amb el fitxer membersFile.

3.2 La funció isLent

Simplement comprova que el llibre no el tingui algun membre. Sabent que representem un llibre que no està en préstec amb el valor -1L solament cal comprovar si no es troba aquest valor.

```
private boolean isLent(Book book){
        return book.getIdMember()!=-1;
    }
```

3.3 La funció canBorrow

La funció canBorrow ja definida en la classe Member, retorna un valor booleà en funció de si el membre pot agafar més llibres. Fa ús de la funció countBooks per comprovar si el nombre de llibres que té el membre es inferior al nombre màxim de llibres MAX_BOOKS.

```
public boolean canBorrow() {
        return countBooks()<MAX_BOOKS;
    }
```

3.3.1 La funció countBooks

De la mateixa manera que comprovàvem si un llibre estava ocupat per un membre observant si el valor del seu ID era diferent de -1L en la funció isLent, la funció countBooks realitza el mateix procés però per totes les posicions possibles del array dels ID dels llibres del membre. En cas de trobar un valor diferent a -1L afegirà al comptador un llibre més.

```
public int countBooks() {
        int count = 0;
        for (long idBook : this.idBooks) {
            if (idBook != -1L) {
                count += 1;
            }
        }
        return count;
    }
```

Amb totes les comprovacions anteriors finalment podem fer el préstec del llibre on afegirem un nou llibre amb la funció addBook al membre, marcarem el ID del membre al llibre i escriurem les modificacions els fitxers binaris, juntament amb el missatge d'èxit al log.

```
book.setIdMember(memberId);
member.addBook(bookId);
booksFile.write(book);
membersFile.write(member);
logFile.ok("Book with tittle: \""+book.getTitle()+"\" lent
successfully to member with id "+ member.getId(), line);
```

En cas contrari, si hagués fallat alguna comprovació escriuria el missatge d'error corresponent.

4. La funció returnBook

De forma contrària a la funció lendBook, returnBook s'encarrega d'eliminar els ID del llibre i del membre. També realitza comprovacions similars:

- **Existeix el llibre:** Comprovar que el ID del llibre existeix.
- **El llibre ja el té un altre membre:** Veure si està ocupat.
- **Existeix el membre:** Comprovar que el ID del membre existeix.
- **El membre té el llibre:** Veure si el membre conté el llibre o no.

Per les tres primeres comprovacions s'utilitzen exactament les mateixes tres funcions auxiliars que a lendBook, solament ens queda comprovar que el membre realment té aquell llibre.

4.1 La funció containsBook

Aquesta funció comprova si donat el ID d'un llibre, el membre té el ID el té associat a ell. Per tant realitzem una simple cerca pel array d'IDs del membre comprovant si els ID coincideixen.

```
    public boolean containsBook(long idBook) {
        for(int i=0;i<MAX_BOOKS;i++){
            if(this.idBooks[i]==idBook){
                return true;
            }
        }
        return false;
    }
```

Si el membre realment té el llibre i totes les comprovacions anteriors s'han complert es procedeix a crear un nou llibre buit amb tots els paràmetres anteriors, i s'elimina el llibre del membre. Finalment s'escriuen les modificacions ens els fitxers binaris i s'envia un missatge d'èxit.

```
book = new Book(book.getId(), book.getTitle());
member.removeBook(bookId);
booksFile.write(book);
membersFile.write(member);
logFile.ok("Book with tittle: \""+book.getTitle()+"\" returned
successfully from member with id: "+member.getId(), line);
```

En cas de no complir-se alguna condició s'escriuria el missatge d'error corresponent.

5. Conclusions

Al llarg de tota la pràctica s'ha treballat amb tots els tipus d'escriptura de fitxers presentats a l'assignatura. La seva implementació conjunta a nivell binari amb la classe PackUtils i fitxers de text han permès mostrar una aplicació directa i funcional del temari treballat.

La implementació de les classes bàsiques com són Member, Book, BooksFile i MmebersFile no han presentat cap tipus de dificultat degut a la senzillesa de les seves funcions i implementacions. Evidentment tenint en compte que s'ha tingut un raonament previ del funcionament de les funcions internes de manipulació de fitxers i els objectius que tenien que realitzar cada funció.

Seguidament com era d'esperar la classe LibraryManager era la que presentava més dificultat al tenir cert nivell de complexitat en les seves funcions. Encara així gràcies al disseny descendent i l'enunciat tant pautat han permès una còmoda implementació de totes el que es demanava, fins i tot una petita aportació pròpia com seria la comprovació d'un membre repetit.

També cal mencionar el mètode a l'hora de comprovar possibles errors, on a les classes bàsiques ja ens aportava els tests del JUnit. En canvi a la classe principal la "dificultat" era comprovar que el resultat en els formats binaris fossin els correctes, per sort en el meu cas no m'he trobat cap dificultat a nivell de codificació dels fitxers, simplement errors petits a l'hora de comprovar cada condició que havia que complir cada comanda, però en general cap complicació tant a nivell de raonament, com de programació o plataforma.

Si es tingués que realitzar novament d'es d'un inici probablement miraria amb més detall com funciona la modificació de fitxers en més profunditat per no completar-la en un inici a "cegues" i per tant, fer-la més còmodament.

Considero que es una bona pràctica pel fet d'integrar tant directament tots els continguts i d'aquesta manera tant original. El que si voldria puntualitzar es el fet d'haver-nos aportat ja feta la classe LogFile, ja que no comportaria més de 5 minuts fer-la però intueixo que s'ha fet d'aquesta manera per comprovar a l'hora de corregir la pràctica el seu correcte funcionament o algun altre motiu. I finalment de les tres pràctiques, aquesta es la que m'ha resultat més fàcil de completar.