PROGRAMMING 3 FINAL PROJECT

PRESENTED BY BĂRBIERU CRINA

CONTENT

)1

02

03

04

05

06

MOTIVATION

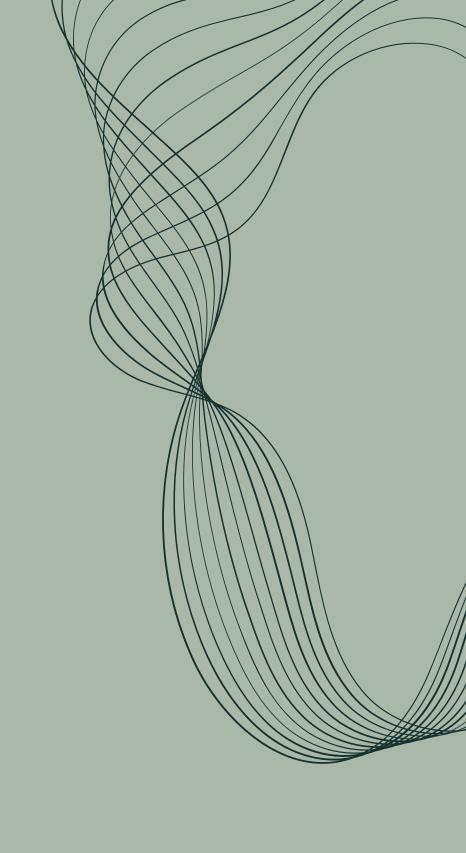
TEST CASES

THREADS

JDBC

USERS

INPUT VALIDATION



MOTIVATION

- Keeping track of books you have read or want to read is difficult:
 - Data is not centralized (between multiple websites, reading lists, etc.)
 - Manual updating is required

reading list

09:02 ESTHER PEREL State Of Affai..



Books read:

14.08.2022 Handwritten note



L.L. WORKS LIBRARY

09:02 https://www.liberatinglibrary.com/

- Updated motivation:
 - Threads: speeds up processes which require searching through lists of books
 - Databases: structured data, efficient retrieval
 - Unit Tests: validating functionalities



UNIT TESTS

	✓ <default package=""></default>	111 ms
	✓ ✓ WishlistBookTest	42 ms
	<pre>test_EmptyConstructor()</pre>	39 ms
	<pre>test_ArgsConstructor()</pre>	3 ms
	✓ ✓ BoughtBookTest	2 ms
	<pre>test_EmptyConstructor()</pre>	1 ms
	<pre>test_ArgsConstructor()</pre>	1 ms
	ApplicationTest	47 ms
	<pre>test_checkDate1()</pre>	42 ms
ı	<pre>test_checkDate2()</pre>	1 ms
ı	<pre>test_checkDate3()</pre>	2 ms
ı	<pre>test_checkDate4()</pre>	1 ms
ı	<pre>test_checkDate5()</pre>	1 ms
	✓ ✓ AuthorTest	2 ms
	<pre>test_EmptyConstructor()</pre>	1 ms
	<pre>test_ArgsConstructor()</pre>	1 ms

- Constructors (with and without arguments) for classes: Author, Book, BoughtBook, WishlistBook, Bookshelf
- Functionalities:
 - check validity of a date
 - find books by author
 - sort books in bookshelf alphabetically

✓ ✓ BookshelfTest	16 ms
<pre>test_findBooksByAuthorA()</pre>	4 ms
test_findBooksByAuthorB()	2 ms
<pre>test_findBooksByAuthorC()</pre>	1 ms
test_EmptyConstructor()	2 ms
<pre>test_sortAlpha1()</pre>	4 ms
test_sortAlpha2()	2 ms
<pre> test_ArgsConstructor() </pre>	1 ms
✓ ✓ BookTest	2 ms
test_EmptyConstructor()	2 ms
<pre>test_ArgsConstructor()</pre>	

```
public void sortAlpha() throws InterruptedException {
    SortThread bt = new SortThread(this.availableBooks);
    SortThread wt = new SortThread(this.wishlistBooks);
    bt.start();
    wt.start();
    public ArrayList<Book>
    bt.join();
    wt.join();
}
```

THREADS

```
A Bookshelf has two components:
```

- Available Books
- Wishlist

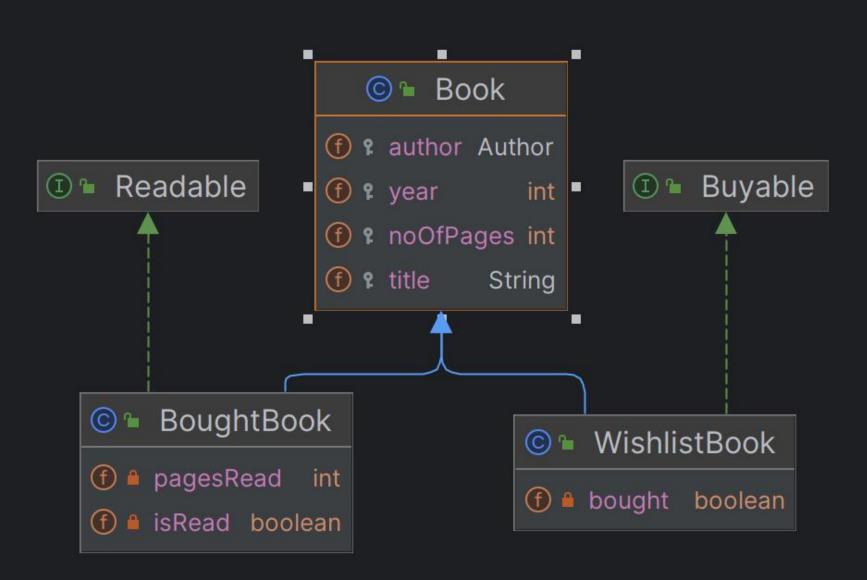
Using threads, we can search for data and perform operation on both lists at the same time.

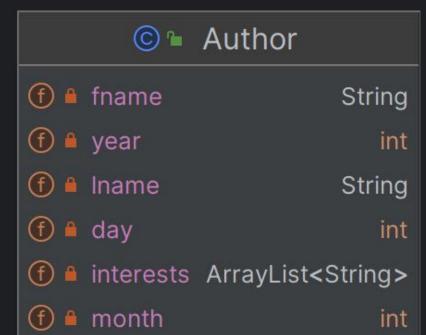
```
public ArrayList<Book> findBooksByAuthor(Author a)
   ArrayList<Book> blist = new ArrayList<>();
   FindAuthorThread bbt = new FindAuthorThread(this.availableBooks, a);
   FindAuthorThread wbt = new FindAuthorThread(this.wishlistBooks, a);
   bbt.start();
   wbt.start();
   try {
        bbt.join();
        wbt.join();
        blist.addAll(bbt.getResult());
        blist.addAll(wbt.getResult());
   return blist;
   } catch (InterruptedException e) {
        throw new RuntimeException(e);
```

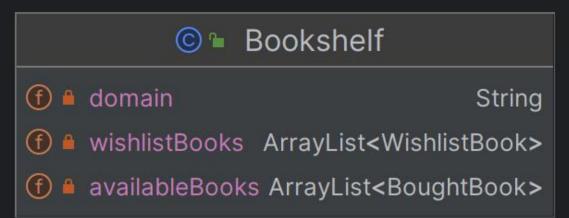
DATABASE CONNECTION



FROM CLASS DIAGRAMS ...







- ♠ Application♠ id InputDevice♠ od OutputDevice
- © InputDevice

 ① is InputStream
 ① scan Scanner
- OutputDeviceos OutputStream
- © ¹ Main
- **⊕** BookNotFoundException

... TO TABLES available_books isbn varchar bigint pagesRead isRead boolean books authors books_bookshelves bookshelves *₽* isbn varchar bigint \wp id Dookshelf_id bigint ₽ id bigint authorid bigint fname varchar isbn varchar varchar name title varchar varchar Iname domain varchar bigint year bigint year bigint pages suggestions wishlist users username varchar isbn varchar varchar username isbn varchar title varchar password varchar boolean bought author_fname varchar !J drawSQL author_Iname bigint

DB QUERIES

Data is loaded from tables to:

- create instances of classes
- avoid duplicates
- validate input

```
/* LOAD BOOKSHELVES FROM DATABASE */
query = " select * from bookshelves; ";
rs = stmt.executeQuery(query);
ArrayList<Bookshelf> bookshelves= new ArrayList<>();
Bookshelf bs;
for(int i=0;i<no0fBookshelves;i++) {</pre>
    rs.next();
    int id = rs.getInt( columnIndex: 1);
    String name = rs.getString(columnIndex: 2);
    bs = new Bookshelf(id, name);
    bookshelves.add(bs);
```

```
/* LOAD BOOKS FOR EACH BOOKSHELF */
String pquery = "Select * from books where isbn in (select isbn from books_bookshelves where
  bookshelf_id = ?);";
PreparedStatement pstmt =conn.prepareStatement(pquery);
String pqAuth = "Select * from authors where id = ?;";
PreparedStatement psauth =conn.prepareStatement(pqAuth);
```

```
/* CHECK IF AUTHOR IS IN DATABASE ALREADY */
if (rs.next()) {
    int id = rs.getInt( columnIndex: 1);
    a.setId(id);
} else {
    String insAuth = "insert into authors(id, fname, lname, year) values (null, ?, ?, ?);";
    PreparedStatement insertAuth = conn.prepareStatement(insAuth);
    insertAuth.setString( parameterIndex: 1, fname);
    insertAuth.setString( parameterIndex: 2, lname);
    insertAuth.setInt( parameterIndex: 3, ayear);
    insertAuth.execute();
```

INSERTING DATA

```
rs = insertBook.executeQuery();
if (rs.next()) {
   od.write(msg: "This book is already in current bookshelf!");
} else {
   String insb2 = "insert into books_bookshelves values(" + bs.getId() + ", " + isbn + ");";
   Statement stmt = conn.createStatement();
   stmt.execute(insb2);
   od.write(msg: "Book has been added successfully!");
}
```

UPDATING DATA

```
while(true)
    try {
     String ans = id.nextLine();
       int pg = Integer.parseInt(ans);
        if(pg<0)
            throw new NumberOutOfRange( msg: "I'm pretty sure you can only read a positive amount of
             pages. Try again:)");
        ((BoughtBook) b).readPages(pg);
        String q = "update available_books set pagesRead="+((BoughtBook) b).getPagesRead()+" , 2
        sisRead = "+((BoughtBook) b).getIsRead() +" where isbn = " + b.getIsbn()+";";
        Statement st =conn.createStatement();
        st.execute(q);
        od.write(String.format("Congrats! Now you have read %d out of %d pages", ((BoughtBook)
         b).getPagesRead(), b.noOfPages));
       break;
```

DELETING DATA

```
if(answer.equals("N"))
{
    sg = "delete from suggestions where isbn = "+isbn+";";
    stmt.execute(sg);
    od.write( msg: "Recommendation has been discarded");
    break;
}
```

```
String isbn = b.getIsbn();
od.write( msg: "Deleting the book...");
bs.deleteBook(b);
String qbooks = "delete from books_bookshelves where isbn = " + b.getIsbn() + " and bookshelf_id=" +bs.getId() + ";";
Statement delbooks = conn.createStatement();
delbooks.execute(gbooks);
od.write( msg: "Book has been deleted successfully from "+bs.getDomain());
```

USERS

OWNER OPTIONS

FRIEND OPTIONS

Please choose one of the options below:

- O.Exit application
- 1.Display available bookshelves
- 2.Display all books from available bookshelves
- 3. Choose a bookshelf to see more details
- 4.Check out suggestions

Please choose one of the options below:

- 0.Exit application
- 1.Display available bookshelves
- 2.Display all books from available bookshelves
- 3. Choose a bookshelf to see more details
- 4.Send a suggestion for your friend

You have chosen Bookshelf of Economics

Please choose one of the options below:

- O.Go back to available bookshelves
- 1.Display current bookshelf
- 2.Display reading progress
- 3. Search a book by title
- 4. Search for an author by name and display their books
- 5.Modify bookshelf
- 6.Sort books alphabetically

You have chosen Bookshelf of Economics

Please choose one of the options below:

- O.Go back to available bookshelves
- 1. Display current bookshelf
- 2.Display reading progress
- 3. Search a book by title
- 4. Search for an author by name and display their books

VALIDATING USER INPUT

```
public boolean checkBookinTable(Connection conn, String isbn, String tableName) throws SQLException {
   boolean result = false;
   String query = "select count(*) from " + tableName +" where isbn = ?;";
   PreparedStatement stmt =conn.prepareStatement(query);
   stmt.setString( parameterIndex: 1, isbn);
   ResultSet rs = stmt.executeQuery();
   if(rs.next() && rs.getInt(columnIndex: 1)>0)
       result = true;
   return result;
                                od.write( msg: "ISBN:");
                                String isbn = id.nextLine();
                                while (isbn.length() != 13 || !isbn.matches( regex: "\\d{13}")) {
                                     od.write( msg: "Please enter a 13-digit ISBN!");
                                     isbn = id.nextLine();
```

