PROGRAMMING 3

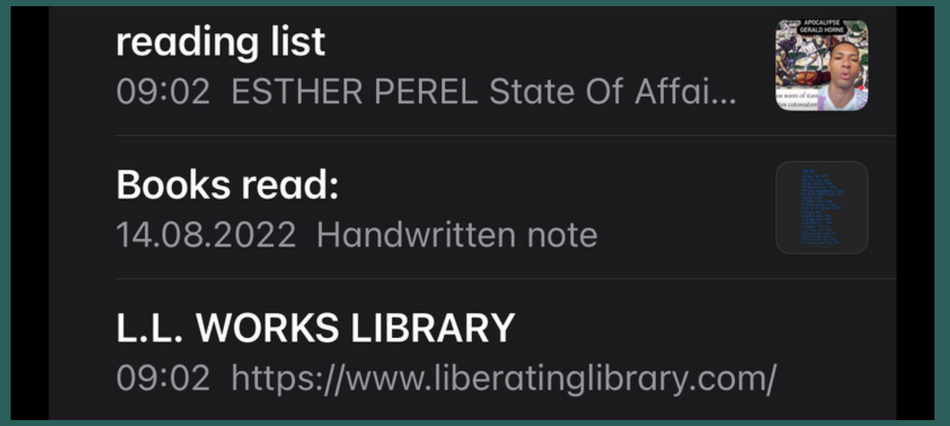
FINAL PROJECT

PRESENTED BY BĂRBIERU CRINA

CONTENT

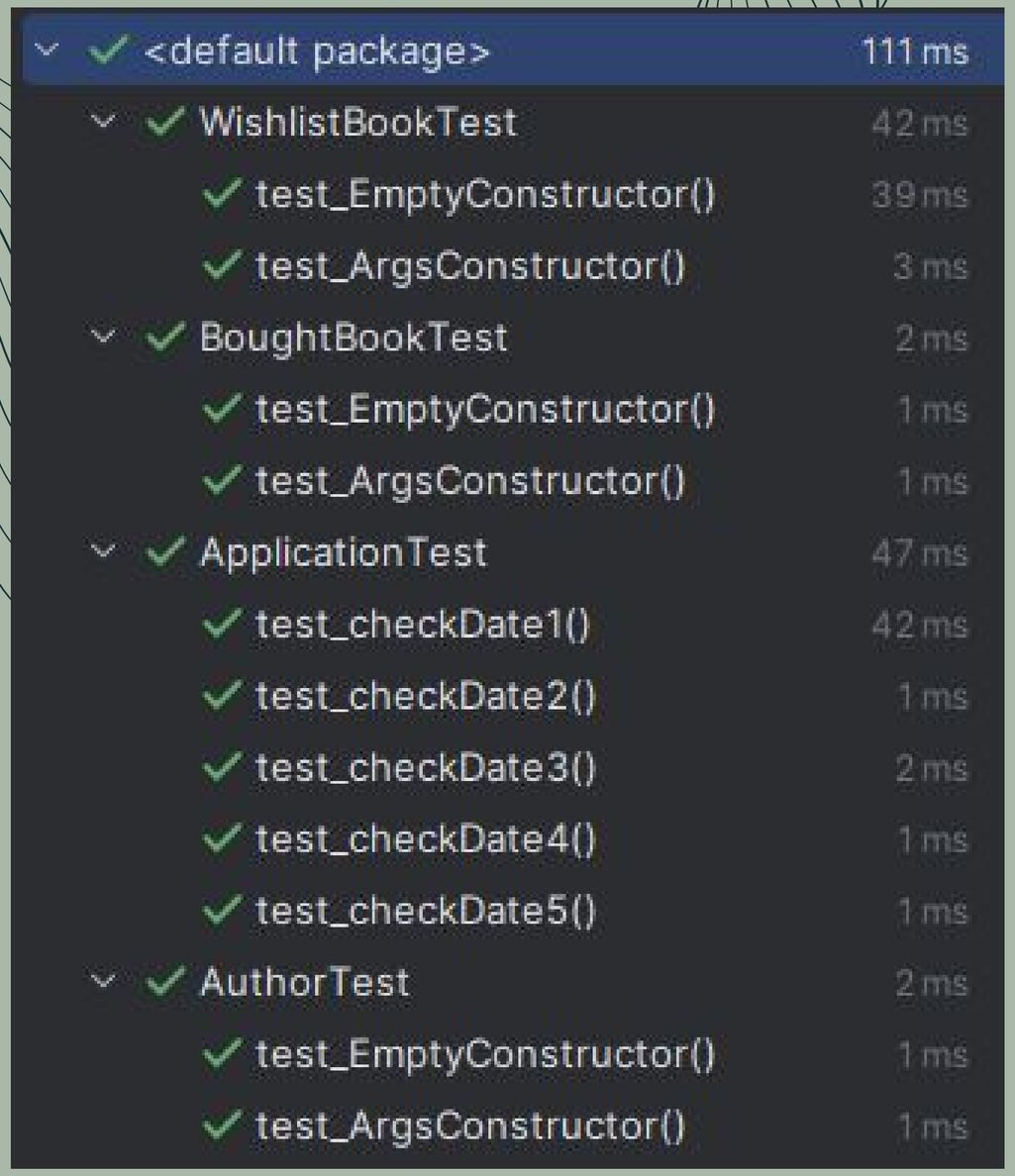
MOTIVATION

- 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

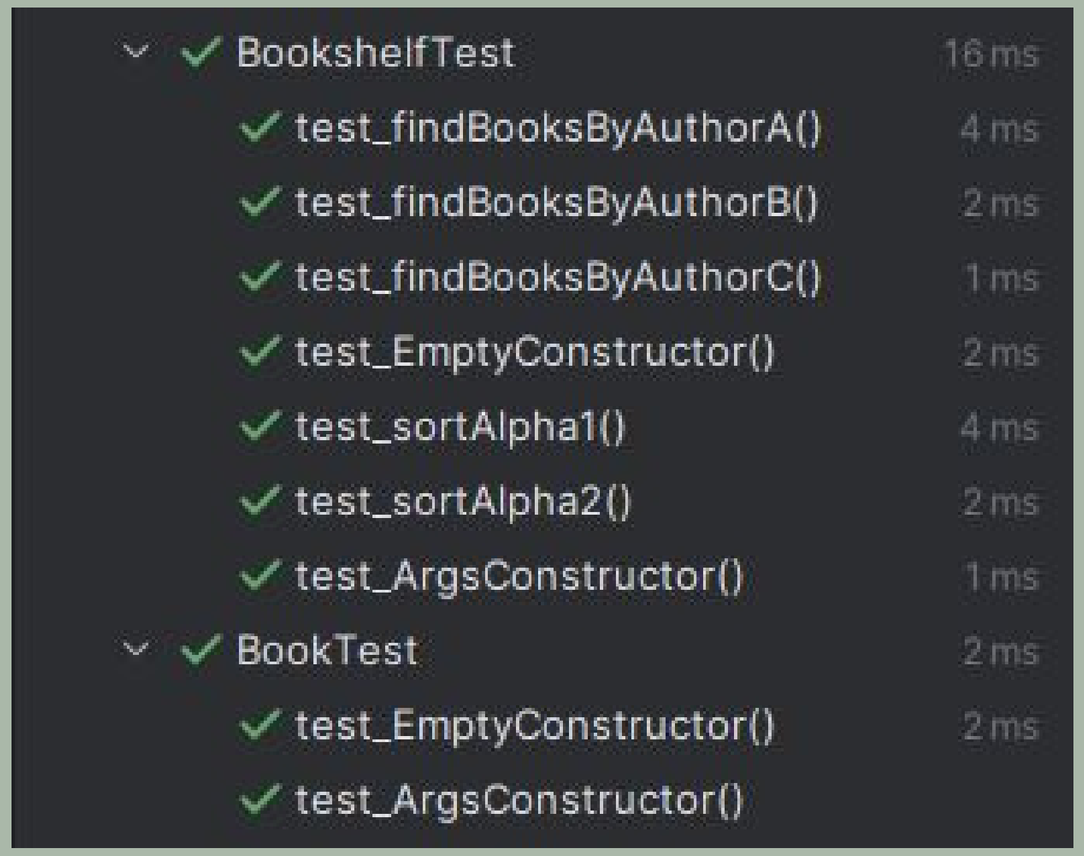


- **Updated motivation:**
  - **Threads: speeds up processes which require searching through lists of books**
  - **Databases: structured data, efficient retrieval**
  - **Unit Tests: validating functionalities**

UNIT TESTS



- **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**



THREADS

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

```java
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);
    }
}
```

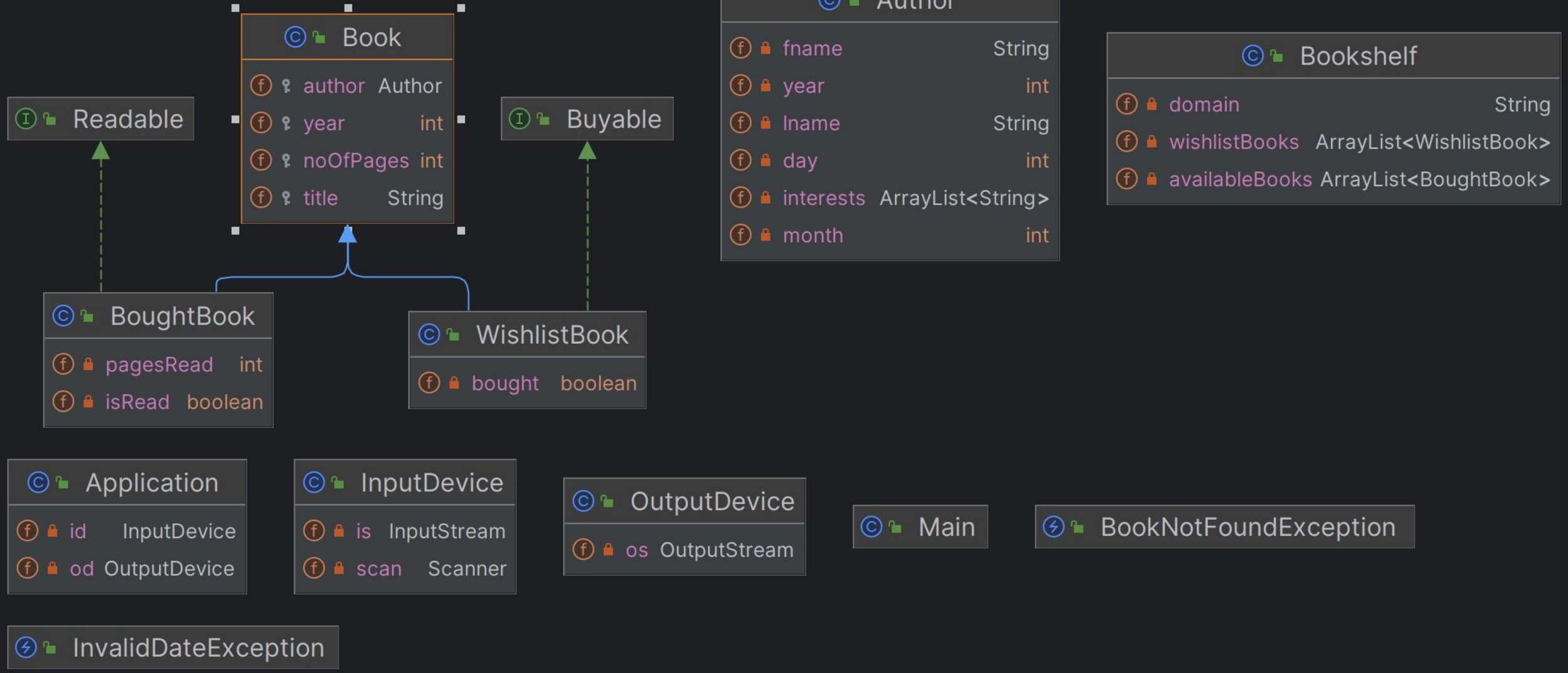
A Bookshelf has two components:

- **Available Books**
- **Wishlist**

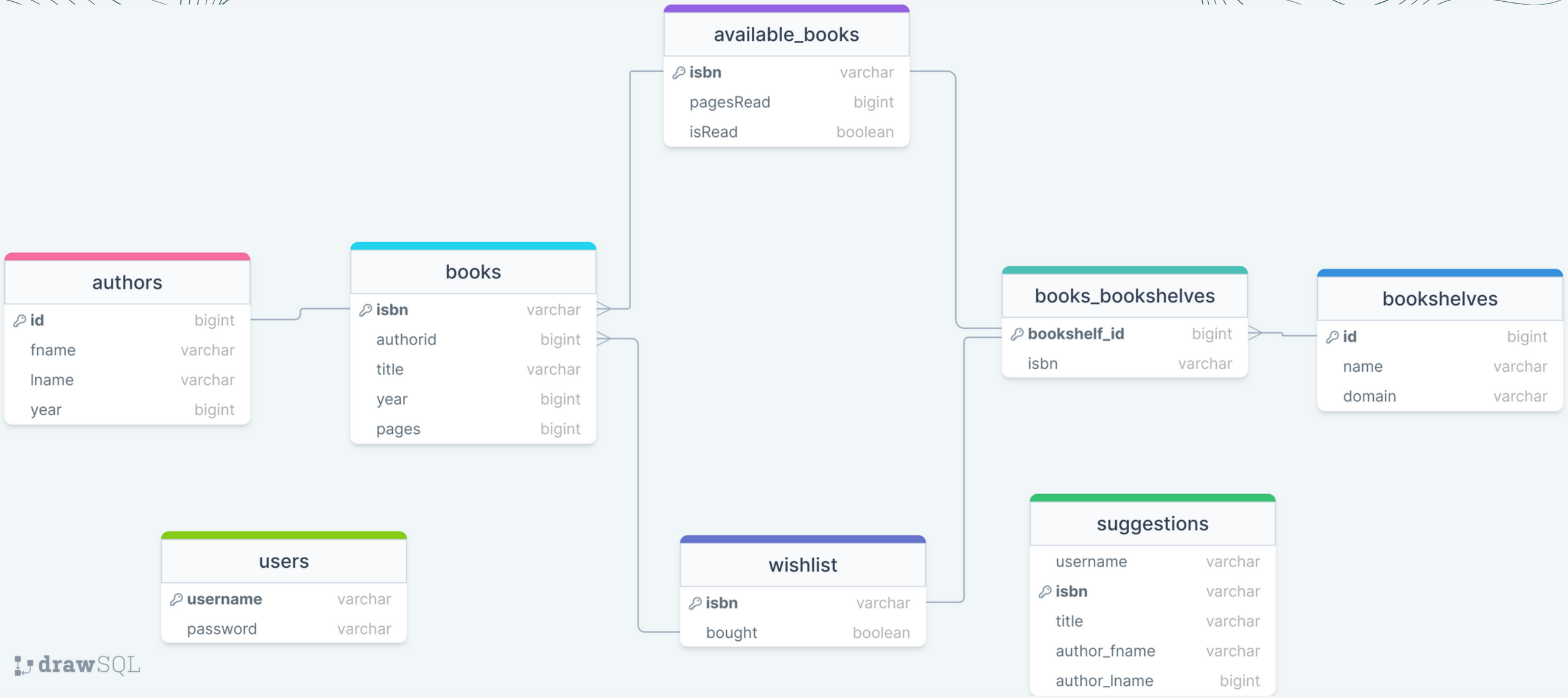
Using threads, we can search for data and perform operation on both lists at the same time.

DATABASE CONNECTION

FROM CLASS DIAGRAMS ...



... TO TABLES



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<noOfBookshelves;i++) {
    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);
```

INSERTING DATA

```
/* 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();
```

```
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()+"
        ,isRead = "+((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(qbooks);
od.write( msg: "Book has been deleted successfully from "+bs.getDomain());
```

USERS

OWNER OPTIONS

```
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.Check out suggestions
```

```
You have chosen Bookshelf of Economics
Please choose one of the options below:
0.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
```

FRIEND OPTIONS

```
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:
0.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();
}
```

THANK YOU