Relational Databases with MySQL Week 10 Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document to the repository. Additionally, push an .sql file with all your queries and your Java project code to the same repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

In this week's coding activity, you will create a menu driven application backed by a MySQL database.

To start, choose one item that you like. It could be vehicles, sports, foods, etc....

Create a new Java project in Eclipse.

Create a SQL script in the project to create a database with one table. The table should be the item you picked.

Write a Java menu driven application that allows you to perform all four CRUD operations on your table.

Tips:

The application does not need to be as complex as the example in the video curriculum.

You need an option for each of the CRUD operations (Create, Read, Update, and Delete).

Remember that PreparedStatment.executeQuery() is only for Reading data and .executeUpdate() is used for Creating, Updating, and Deleting data.

Remember that both parameters on PreparedStatements and the ResultSet columns are based on indexes that start with 1, not 0.

Screenshots of Code:

Menu.java | DBConnection.java | db_MyBooks.sql | PublishedBookMenu.java | Bool

src > application > Menu.java > Menu > start()

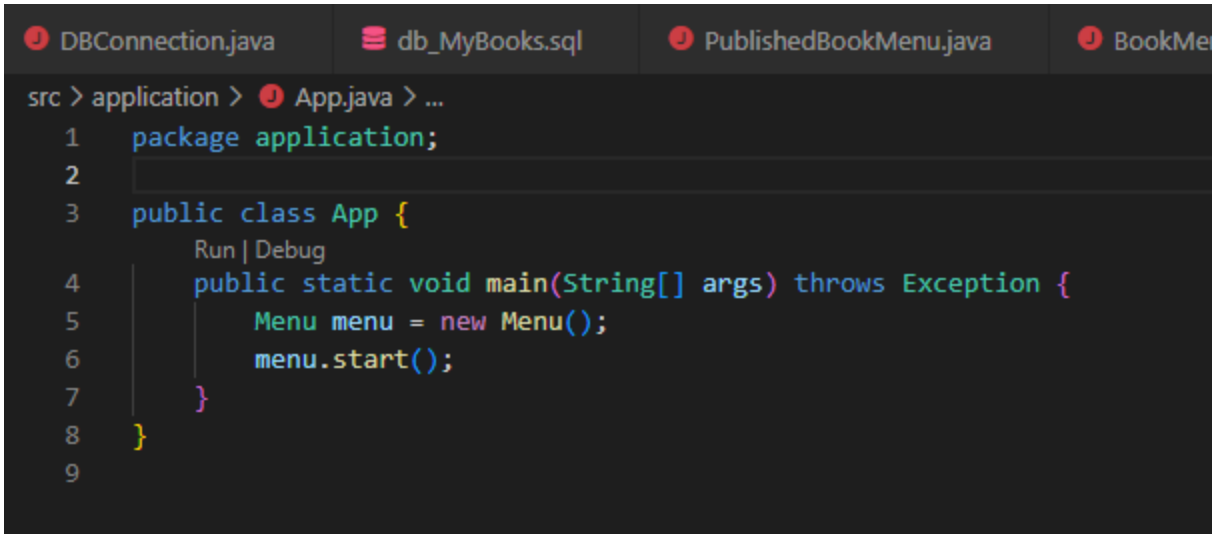
```java
package application;

import java.sql.SQLException;
import java.util.List;
import java.util.Scanner;

import dao.PublishedBookDao;
import entity.PublishedBook;

public class Menu {
    private int choice;
    private PublishedBookDao publishedBookDao = new PublishedBookDao();
    private Scanner kb = new Scanner(System.in);
    private List<PublishedBook> allPBooks;

    public void start(){
        boolean end = false;
        AuthorMenu authorMenu = new AuthorMenu();
        AddPBookMenu addPBookMenu = new AddPBookMenu();
        PublishedBookMenu publishedBookMenu = new PublishedBookMenu();
        BookMenu bookMenu = new BookMenu();

        while (!end){
            showMainMenu();
            getInput();

            try {
                switch (choice) {
                    case 1:
                        getAllPBooks();
                        break;
                    case 2:
                        authorMenu.showAllAuthors();
                        break;
                    case 3:
                        bookMenu.bookVersions();;
                        break;
                    case 4:
                        addPBookMenu.addBook();
                        break;
                    case 5:
                        publishedBookMenu.finishBook();
                        break;
                    case 6:
                        publishedBookMenu.deleteBook();
                        break;
```

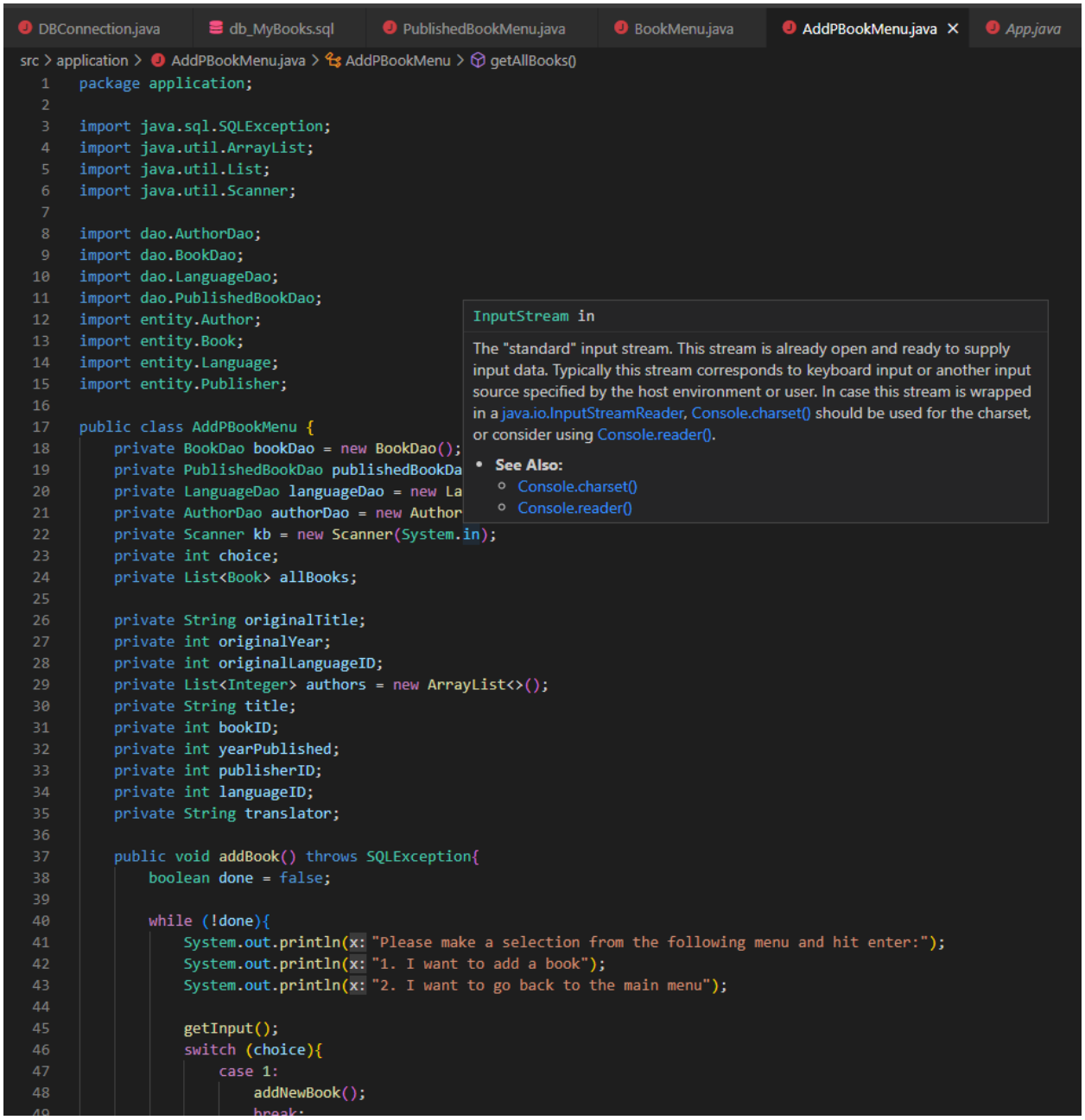
Menu.java | DBConnection.java | db_MyBooks.sql | PublishedBookMenu.java | BookMenu.java | AddPBookMe

src > application > Menu.java > Menu > start()

```
46                          break;
47                      case 7:
48                          kb.close();
49                          end = true;
50                          break;
51                      default:
52                          System.out.println(x: "That was not a valid choice please try again.");
53                  }
54              } catch (Exception e) {
55                  System.out.println(x: "something went wrong");
56                  System.out.println(e.getMessage());
57                  System.out.println(e.getStackTrace());
58                  System.out.println(e.toString());
59          }
60          }
61
62          kb.close();
63
64      }//start
65
66      private void showMainMenu(){
67          System.out.println();
68          System.out.println(x: "Please select one of the following options and then hit enter: ");
69          System.out.println(x: "1. See all books in your library.");
70          System.out.println(x: "2. See all authors in your library.");
71          System.out.println(x: "3. See how many versions of a specific book you have.");
72          System.out.println(x: "4. Add a new book.");
73          System.out.println(x: "5. Set a book to finished.");
74          System.out.println(x: "6. Remove a book from your library.");
75          System.out.println(x: "7. Exit.");
76          System.out.println();
77      }//showMainMenu
78
79      private void getAllPBooks() throws SQLException{
80          allPBooks = publishedBookDao.getAllBooks();
81          for (int i = 1; i <= allPBooks.size(); i++){
82              System.out.print(i + ". ");
83              allPBooks.get(i-1).printDetails();
84              System.out.println();
85          }
86      }//getAllPBooks
87
88      private int getInput(){
89          try {
90              choice = Integer.parseInt(kb.nextLine());
91          } catch (Exception e) {
92              choice = 0;
93          }
94          return choice;
```

```
92                  choice = 0;
93              }
94              return choice;
95          }
96      }
97
```


DBConnection.java
db_MyBooks.sql
PublishedBookMenu.java
BookMe
src > application > App.java > ...
1 package application;
2
3 public class App {
Run | Debug
4 public static void main(String[] args) throws Exception {
5 Menu menu = new Menu();
6 menu.start();
7 }
8 }
9


DBConnection.java
db_MyBooks.sql
PublishedBookMenu.java
BookMenu.java
AddPBookMenu.java
App.java
src > application > AddPBookMenu.java > AddPBookMenu > getAllBooks()
package application;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import dao.AuthorDao;
import dao.BookDao;
import dao.LanguageDao;
import dao.PublishedBookDao;
import entity.Author;
import entity.Book;
import entity.Language;
import entity.Publisher;
public class AddPBookMenu {
private BookDao bookDao = new BookDao();
private PublishedBookDao publishedBookDa
private LanguageDao languageDao = new La
private AuthorDao authorDao = new Author
private Scanner kb = new Scanner(System.in);
private int choice;
private List<Book> allBooks;
private String originalTitle;
private int originalYear;
private int originalLanguageID;
private List<Integer> authors = new ArrayList<>();
private String title;
private int bookID;
private int yearPublished;
private int publisherID;
private int languageID;
private String translator;
public void addBook() throws SQLException{
boolean done = false;
while (!done){
System.out.println(x: "Please make a selection from the following menu and hit enter:");
System.out.println(x: "1. I want to add a book");
System.out.println(x: "2. I want to go back to the main menu");
getInput();
switch (choice){
case 1:
addNewBook();
InputStream in
The "standard" input stream. This stream is already open and ready to supply input data. Typically this stream corresponds to keyboard input or another input source specified by the host environment or user. In case this stream is wrapped in a java.io.InputStreamReader, Console.charset() should be used for the charset, or consider using Console.reader().
See Also:
Console.charset()
Console.reader()

src > application > AddPBookMenu.java > AddPBookMenu > getAllBooks()

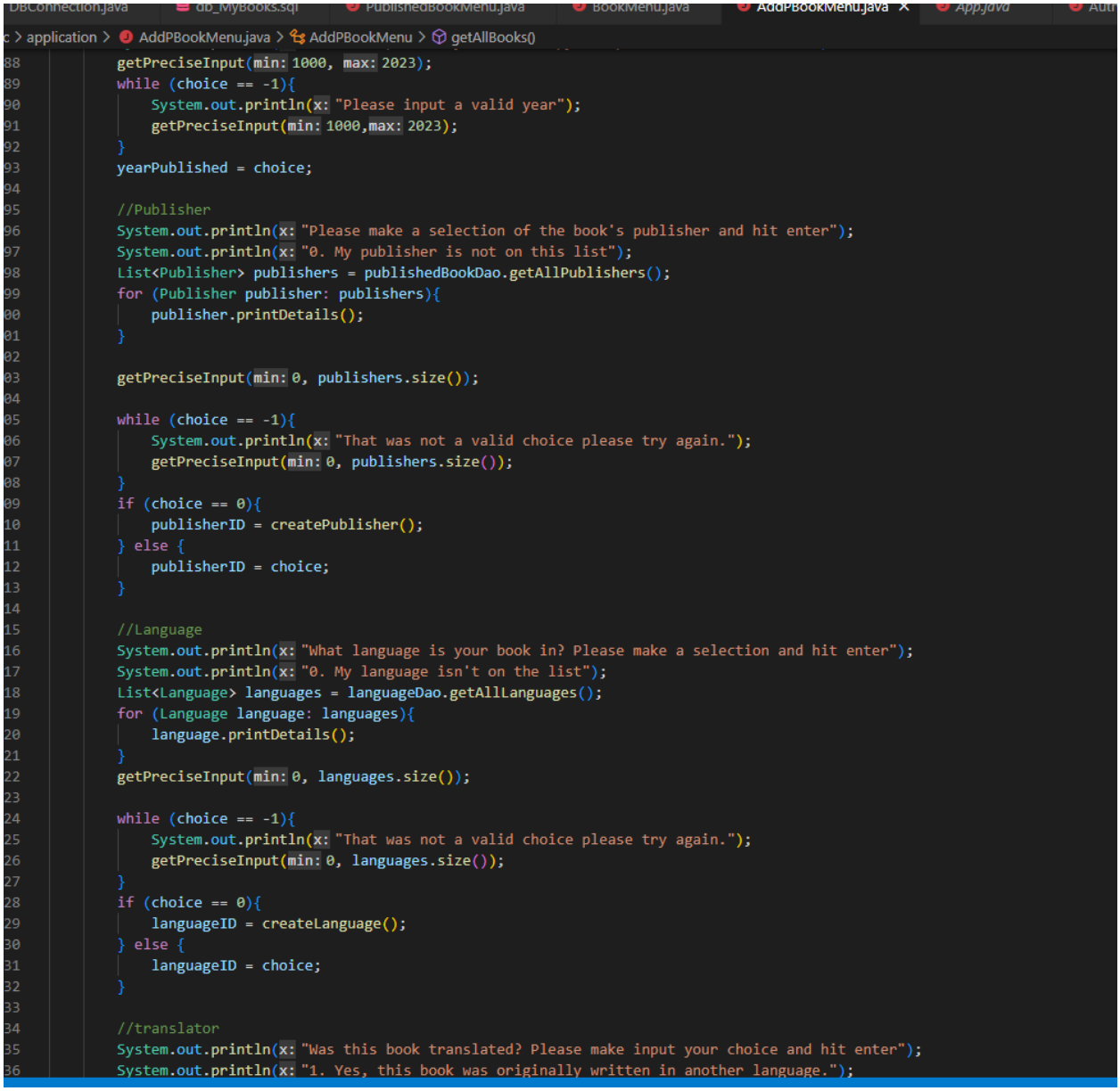
```
43              System.out.println(x: "2. I want to go back to the main menu");
44
45              getInput();
46              switch (choice){
47                  case 1:
48                      addNewBook();
49                      break;
50                  case 2:
51                      done = true;
52                      return;
53                  default:
54                      System.out.println(x: "That was not a valid choice please try again.");
55              }
56          }
57      }
58
59      private void addNewBook() throws SQLException{
60          System.out.println(x: "We're going to start with information about the original edition of this book");
61          System.out.println(x: "This information is usually foind on the first couple of pages of a book");
62          System.out.println(x: "Please select 0 if your book is not one of these books or select the book's ID to add a new version");
63          System.out.println(x: "0. This is a completely new book");
64          getAllBooks();
65          choice = getPreciseInput(min: 0, allBooks.size());
66          while (choice == -1){
67              System.out.println(x: "That was not a valid choice please try again.");
68              choice = getPreciseInput(min: 0, allBooks.size());
69          }
70          if (choice == 0){
71              createBook();
72          } else {
73              createPublishedBook(choice);
74          }
75      }
76
77      private void createBook() throws SQLException{
78          //original title
79          System.out.println(x: "Please input the original title of the book when it was first published");
80          System.out.println(x: "the original title might be in a different language if the book has been translated");
81          System.err.println(x: "please hit enter when you are done");
82          originalTitle = kb.nextLine();
83
84          //original year
85          System.out.println(x: "Please enter the year when the book was first published");
86          getPreciseInput(min: 1000, max: 2023);
87          while (choice == -1){
88              System.out.println(x: "That was not a valid choice please try again.");
89              getPreciseInput(min: 1000, max: 2023);
90          }
91          originalYear = choice;
```

DBConnection.java | db_MyBooks.sql | PublishedBookMenu.java | BookMenu.java | AddPBookMenu.java | App.java | AuthorM

src > application > AddPBookMenu.java > AddPBookMenu > getAllBooks()

```
91          originalYear = choice;
92
93          //original language
94          System.out.println(x: "What language was the first edition printed in? Please make a selection and hit enter");
95          System.out.println(x: "0. My language isn't on the list");
96          List<Language> languages = languageDao.getAllLanguages();
97          for (Language language: languages){
98              language.printDetails();
99          }
100         getPreciseInput(min: 0, languages.size());
101
102         while (choice == -1){
103             System.out.println(x: "That was not a valid choice please try again.");
104             getPreciseInput(min: 0, languages.size());
105         }
106         if (choice == 0){
107             originalLanguageID = createLanguage();
108         } else {
109             originalLanguageID = choice;
110         }
111
112         createAuthors();
113         if (originalLanguageID != 0 && authors.size() != 0){
114             bookID = bookDao.createBook(originalTitle, originalYear, originalLanguageID, authors);
115             System.out.println();
116             createPublishedBook(bookID);
117         } else {
118             System.out.println(x: "something went wrong, please try again");
119             return;
120         }
121
122     }
123
124     private void createAuthors() throws SQLException{
125         boolean done = false;
126
127
128         while (!done){
129             System.out.println(x: "Please input who wrote your book");
130             System.out.println(x: "If your book has multiple authors please input one at a time");
131
132             System.out.println(x: "0. My author is not on the list");
133             List<Author> existingAuthors = authorDao.getAllAuthors();
134             for(Author author:existingAuthors){
135                 author.printDetails();
136             }
137             System.err.println((existingAuthors.size() + 1) + ". I am done adding authors");
138             getPreciseInput(min: 0, (existingAuthors.size()+1));
```

```java
            getPreciseInput(min: 0, (existingAuthors.size()+1));
            while (choice == -1){
                System.out.println(x: "That was not a valid choice please try again.");
                getPreciseInput(min: 0, (existingAuthors.size()+1));
            }
            if (choice == 0){
                authors.add(addAuthor2DB());
                System.out.println(authors);
            } else if (choice == (existingAuthors.size()+1) && authors.size() > 0){
                done = true;
            } else if (choice == (existingAuthors.size()+1) && authors.size() == 0){
                System.out.println(x: "you need to add at least one author");
            } else {
                authors.add(choice);
            }
        }
    }

    private int addAuthor2DB() throws SQLException {
        System.out.println(x: "please input the author's first name and hit enter");
        String firstName = kb.nextLine();
        while (firstName.equals(anObject: "")){
            System.out.println(x: "the first name cannot be blank");
            firstName = kb.nextLine();
        }

        System.out.println(x: "please input the author's middle name, if they have none just hit enter");
        String middleName = kb.nextLine();

        System.out.println(x: "please input the author's last name, if they have none just hit enter");
        String lastName = kb.nextLine();

        System.out.println(x: "Some authors use a nom de plume if the author also goes by a different name please put that in");
        System.out.println(x: "if the author does not have another name that they go by please just hit enter");
        String aka = kb.nextLine();

        return authorDao.addAuthor(firstName, middleName, lastName, aka);
    }



    private void createPublishedBook (int bookID) throws SQLException{
        System.out.println(x: "Now we need information about this specific copy:");

        //title
        System.out.println(x: "Please input the title of this book and hit enter");
        title = kb.nextLine();
```


DBConnection.java
db_MyBooks.sql
PublishedBookMenu.java
BookMenu.java
AddPBookMenu.java
App.java
c > application > AddPBookMenu.java > AddPBookMenu > getAllBooks()
getPreciseInput(min: 1000, max: 2023);
while (choice == -1){
System.out.println(x: "Please input a valid year");
getPreciseInput(min: 1000,max: 2023);
}
yearPublished = choice;
//Publisher
System.out.println(x: "Please make a selection of the book's publisher and hit enter");
System.out.println(x: "0. My publisher is not on this list");
List<Publisher> publishers = publishedBookDao.getAllPublishers();
for (Publisher publisher: publishers){
publisher.printDetails();
}
getPreciseInput(min: 0, publishers.size());
while (choice == -1){
System.out.println(x: "That was not a valid choice please try again.");
getPreciseInput(min: 0, publishers.size());
}
if (choice == 0){
publisherID = createPublisher();
} else {
publisherID = choice;
}
//Language
System.out.println(x: "What language is your book in? Please make a selection and hit enter");
System.out.println(x: "0. My language isn't on the list");
List<Language> languages = languageDao.getAllLanguages();
for (Language language: languages){
language.printDetails();
}
getPreciseInput(min: 0, languages.size());
while (choice == -1){
System.out.println(x: "That was not a valid choice please try again.");
getPreciseInput(min: 0, languages.size());
}
if (choice == 0){
languageID = createLanguage();
} else {
languageID = choice;
}
//translator
System.out.println(x: "Was this book translated? Please make input your choice and hit enter");
System.out.println(x: "1. Yes, this book was originally written in another language.");

DBConnection.java | db_MyBooks.sql | PublishedBookMenu.java | BookMenu.java | AddPBookMenu.java | App.java | Autho

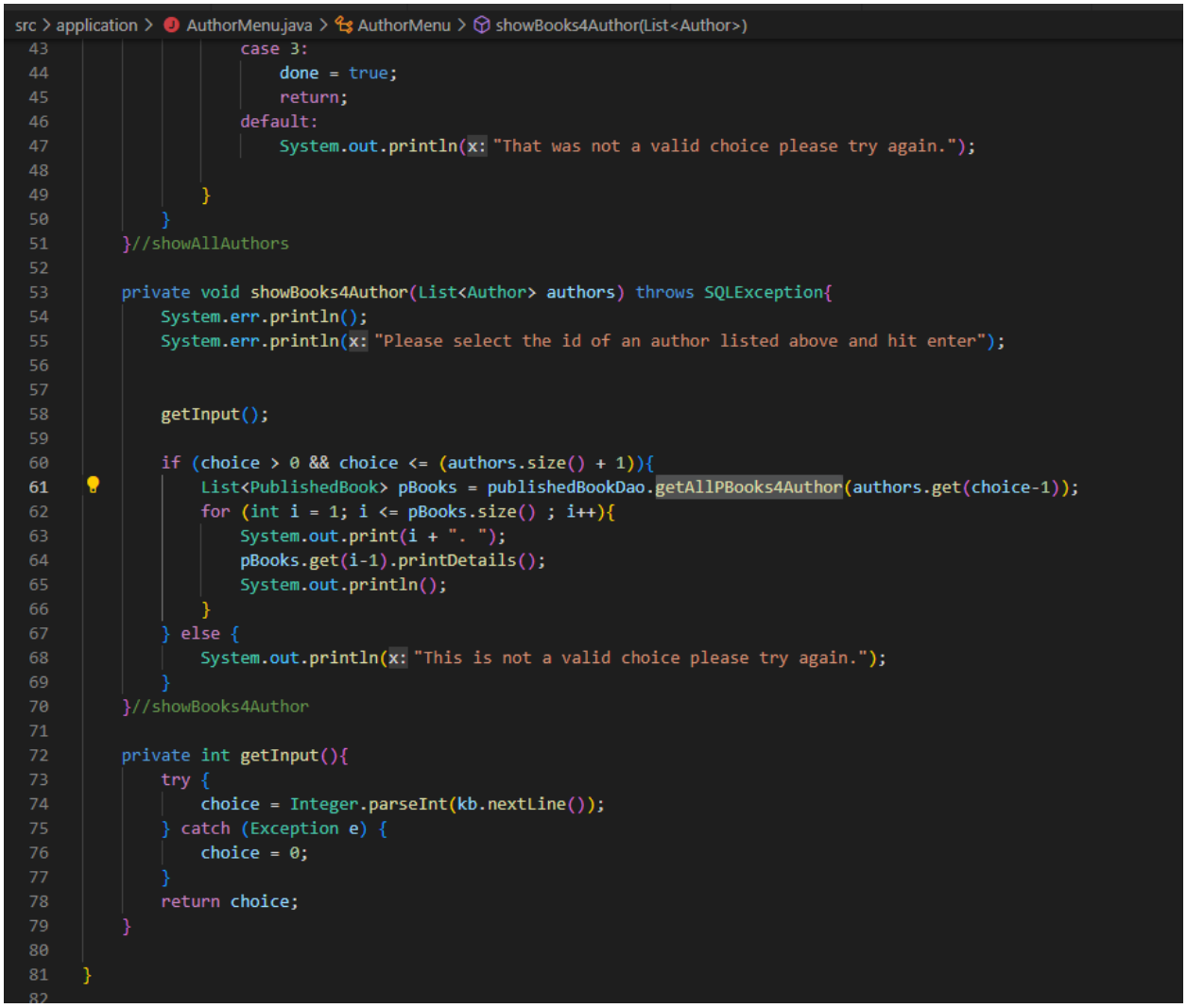
src > application > AddPBookMenu.java > AddPBookMenu > getAllBooks()

```
233
234         //translator
235         System.out.println(x: "Was this book translated? Please make input your choice and hit enter");
236         System.out.println(x: "1. Yes, this book was originally written in another language.");
237         System.out.println(x: "2. No, this book is in it's original language");
238         getPreciseInput(min: 1, max: 2);
239         while (choice == -1){
240             System.out.println(x: "That was not a valid choice please try again.");
241             getPreciseInput(min: 1, max: 2);
242         }
243         if (choice == 1){
244             System.err.println(x: "Please input the translator's name in and hit enter.");
245             translator = kb.nextLine();
246         } else {
247             translator = null;
248         }
249         if(bookID != 0 && publisherID != 0 && languageID != 0){
250             int newpbookID = publishedBookDao.addPBook(bookID, title, yearPublished, publisherID, languageID, translator);
251             if (newpbookID != 0){
252                 System.out.println(x: "SUCCESS your book has been added");
253             }
254         } else {
255             System.out.println(x: "something went wrong please try again");
256             return;
257         }
258     }//Add New Published book
259
260     private int createLanguage() throws SQLException{
261         System.out.println(x: "Please input the language you'd like to input and hit enter");
262         String newLanguage = kb.nextLine();
263         int newLanguageID = languageDao.addLanguage(newLanguage);
264         return newLanguageID;
265     }
266
267     private int createPublisher() throws SQLException{
268         System.out.println(x: "Please input the publisher you'd like to input and hit enter");
269         String newPublisher = kb.nextLine();
270         int newPublisherID = publishedBookDao.addPublisher(newPublisher);
271         return newPublisherID;
272     }
273
274     private void getAllBooks() throws SQLException{
275         allBooks = bookDao.getAllBooks();
276         for (Book book: allBooks){
277             book.printDetails();
278         }
279     }//getAllBooks
```

```
280
281     private void getInput(){
282         try {
283             choice = Integer.parseInt(kb.nextLine());
284         } catch (Exception e) {
285             choice = -1;
286         }
287     }
288
289     private int getPreciseInput(int min, int max){
290         try {
291             choice = Integer.parseInt(kb.nextLine());
292             if (choice < min || choice > max){ choice = -1;}
293         } catch (Exception e) {
294             choice = -1;
295         }
296         return choice;
297     }
298
299 }
300
```

src > application > AuthorMenu.java > AuthorMenu > showBooks4Author(List<Author>)

```java
package application;

import java.sql.*;
import java.util.List;
import java.util.Scanner;

import dao.AuthorDao;
import dao.PublishedBookDao;
import entity.Author;
import entity.PublishedBook;

public class AuthorMenu {
    private AuthorDao authorDao = new AuthorDao();
    private PublishedBookDao publishedBookDao = new PublishedBookDao();
    private int choice;
    private Scanner kb = new Scanner(System.in);

    public void showAllAuthors() throws SQLException{
        boolean done = false;
        List<Author> authors = authorDao.getAllAuthors();
        for (Author author : authors){
            author.printDetails();
        }

        while (!done){
            System.out.println();
            System.out.println(x: "Please choose one of the following options and hit enter:");
            System.out.println(x: "1. To see all the books in your library by a specific author");
            System.out.println(x: "2. Show all authors again");
            System.out.println(x: "3. Go back");

            getInput();

            switch (choice){
                case 1:
                    showBooks4Author(authors);
                    break;
                case 2:
                    for (Author author : authors){
                        author.printDetails();
                    }
                    break;
                case 3:
                    done = true;
                    return;
                default:
                    System.out.println(x: "That was not a valid choice please try again.");

```


src > application > AuthorMenu.java > AuthorMenu > showBooks4Author(List<Author>)
43 case 3:
44 done = true;
45 return;
46 default:
47 System.out.println(x: "That was not a valid choice please try again.");
48
49 }
50 }
51 }//showAllAuthors
52
53 private void showBooks4Author(List<Author> authors) throws SQLException{
54 System.err.println();
55 System.err.println(x: "Please select the id of an author listed above and hit enter");
56
57
58 getInput();
59
60 if (choice > 0 && choice <= (authors.size() + 1)){
61 List<PublishedBook> pBooks = publishedBookDao.getAllPBooks4Author(authors.get(choice-1));
62 for (int i = 1; i <= pBooks.size() ; i++){
63 System.out.print(i + ". ");
64 pBooks.get(i-1).printDetails();
65 System.out.println();
66 }
67 } else {
68 System.out.println(x: "This is not a valid choice please try again.");
69 }
70 }//showBooks4Author
71
72 private int getInput(){
73 try {
74 choice = Integer.parseInt(kb.nextLine());
75 } catch (Exception e) {
76 choice = 0;
77 }
78 return choice;
79 }
80
81 }
82

src > application > BookMenu.java > BookMenu > bookVersions()

```java
package application;

import java.sql.SQLException;
import java.util.List;
import java.util.Scanner;

import dao.BookDao;
import dao.PublishedBookDao;
import entity.Book;
import entity.PublishedBook;

public class BookMenu {

    private List<Book> allBooks;
    private List<PublishedBook> pBooks;
    private BookDao bookDao = new BookDao();
    private PublishedBookDao publishedBookDao = new PublishedBookDao();
    private int choice;
    private Scanner kb = new Scanner(System.in);


    public void bookVersions() throws SQLException{
        System.out.println(x: "Which book do you want to see all the versions of?");
        System.out.println(x: "0. go back to the main menu");
        getAllBooks();
        choice = getPreciseInput(min: 0, allBooks.size());
        while (choice == -1){
            choice = getPreciseInput(min: 0, allBooks.size());
        }
        if (choice == 0){
            return;
        } else {
            int bookID = choice;
            pBooks = publishedBookDao.getAllPBooks4BookID(bookID);
            if (pBooks.size() == 0){
                System.err.println(x: "you don't have any versions of this book in your library");
            } else {
                for(int i = 1; i <= pBooks.size(); i++){
                    System.out.print(i + ". ");
                    pBooks.get(i-1).printDetails();
                    System.out.println();
                }
            }
        }



    }//bookversions
```

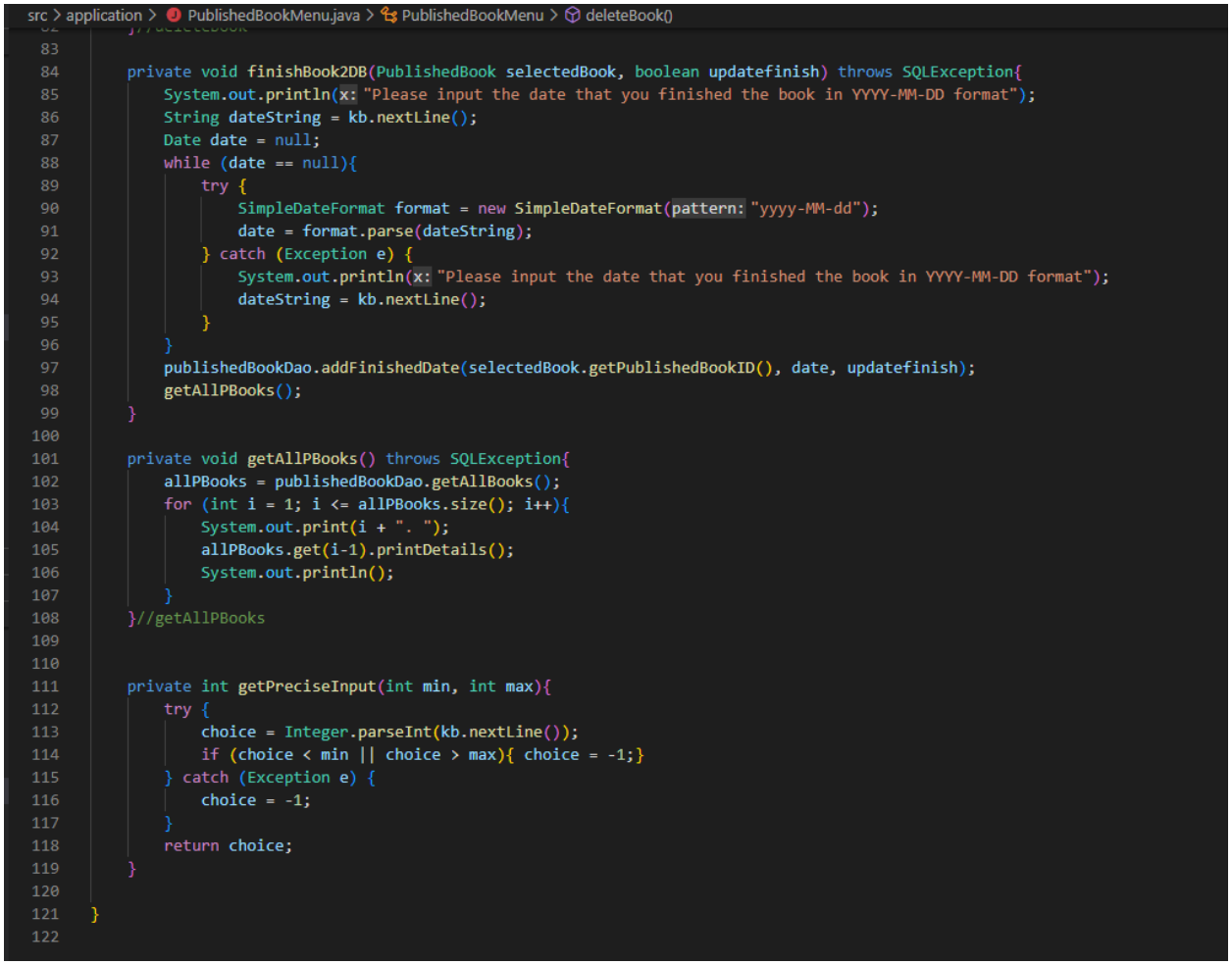
```
50      private void getAllBooks() throws SQLException {
51          allBooks = bookDao.getAllBooks();
52          for (Book book: allBooks){
53              book.printDetails();
54          }
55      }//getAllBooks
56
57      private int getPreciseInput(int min, int max){
58          try {
59              choice = Integer.parseInt(kb.nextLine());
60              if (choice < min || choice > max){ choice = -1;}
61          } catch (Exception e) {
62              choice = -1;
63          }
64          return choice;
65      }//get precise input
66
67
68  }
69
```

src > application > PublishedBookMenu.java > PublishedBookMenu > deleteBook()

```java
package application;

import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.Scanner;

import dao.PublishedBookDao;
import entity.PublishedBook;

public class PublishedBookMenu {

    private dao.PublishedBookDao lPBooks;
    private PublishedBookDao publishedBookDao = new PublishedBookDao();
    private Scanner kb = new Scanner(System.in);
    private int choice;

    public void finishBook() throws SQLException{
        int choice;
        System.out.println(x: "Please select which book you would like to set to finished");
        System.out.println(x: "0. go back to the main menu");
        getAllPBooks();
        choice = getPreciseInput(min: 0, allPBooks.size());
        while (choice == -1){
            choice = getPreciseInput(min: 0, allPBooks.size());
        }
        PublishedBook selectedBook = allPBooks.get(choice-1);
        if(selectedBook.isFinished()){
            System.out.println(x: "you already finished this book");
            System.out.println(x: "we are currently unable to add multiple finishdates to one book");
            System.out.println(x: "do you want to update the old date to the new date?");
            System.out.println(x: "1. yes, please update to the new date");
            System.out.println(x: "2. no, please take me back to the main menu");
            choice = getPreciseInput(min: 1, max: 2);
            while (choice == -1){
                choice = getPreciseInput(min: 1, max: 2);
            }
            if (choice == 1){
                finishBook2DB(selectedBook, updatefinish: true);
            } else if (choice == 2){
                return;
            }
        } else {
            finishBook2DB(selectedBook, updatefinish: false);
        }

    }//finishbook
```

src > application > PublishedBookMenu.java > PublishedBookMenu > deleteBook()

```
48      }//finishbook
49
50      public void deleteBook() throws SQLException{
51          System.out.println(x: "Please select which book you would like to delete and hit enter");
52          System.out.println(x: "0. I don't want to delete any books, take me back to the main menu");
53          getAllPBooks();
54
55          int choice = getPreciseInput(min: 0, allPBooks.size());
56          while (choice == -1){
57              System.out.println(x: "That was not a valid choice please try again.");
58              choice = getPreciseInput(min: 0, allPBooks.size());
59          }
60          if (choice == 0){
61              return;
62          } else {
63              PublishedBook selectedBook = allPBooks.get(choice - 1);
64
65              System.out.println(x: "are you sure you want to delete this book?");
66              selectedBook.printDetails();
67              System.out.println();
68              System.out.println(x: "1. yes, delete it already!");
69              System.out.println(x: "2. no, let's go back to safety!");
70              choice = getPreciseInput(min: 1, max: 2);
71              while (choice == -1){
72                  choice = getPreciseInput(min: 1, max: 2);
73              }
74              if (choice == 1){
75                  publishedBookDao.deletePBook(selectedBook.getPublishedBookID());
76                  getAllPBooks();
77                  return;
78              } else {
79                  return;
80              }
81          }
82      }//deletebook
83
84      private void finishBook2DB(PublishedBook selectedBook, boolean updatefinish) throws SQLException{
85          System.out.println(x: "Please input the date that you finished the book in YYYY-MM-DD format");
86          String dateString = kb.nextLine();
87          Date date = null;
88          while (date == null){
89              try {
90                  SimpleDateFormat format = new SimpleDateFormat(pattern: "yyyy-MM-dd");
91                  date = format.parse(dateString);
92              } catch (Exception e) {
93                  System.out.println(x: "Please input the date that you finished the book in YYYY-MM-DD format");
94                  dateString = kb.nextLine();
95              }
96          }
```


src > application > PublishedBookMenu.java > PublishedBookMenu > deleteBook()
83
84 private void finishBook2DB(PublishedBook selectedBook, boolean updatefinish) throws SQLException{
85 System.out.println(x: "Please input the date that you finished the book in YYYY-MM-DD format");
86 String dateString = kb.nextLine();
87 Date date = null;
88 while (date == null){
89 try {
90 SimpleDateFormat format = new SimpleDateFormat(pattern: "yyyy-MM-dd");
91 date = format.parse(dateString);
92 } catch (Exception e) {
93 System.out.println(x: "Please input the date that you finished the book in YYYY-MM-DD format");
94 dateString = kb.nextLine();
95 }
96 }
97 publishedBookDao.addFinishedDate(selectedBook.getPublishedBookID(), date, updatefinish);
98 getAllPBooks();
99 }
100
101 private void getAllPBooks() throws SQLException{
102 allPBooks = publishedBookDao.getAllBooks();
103 for (int i = 1; i <= allPBooks.size(); i++){
104 System.out.print(i + ". ");
105 allPBooks.get(i-1).printDetails();
106 System.out.println();
107 }
108 }//getAllPBooks
109
110
111 private int getPreciseInput(int min, int max){
112 try {
113 choice = Integer.parseInt(kb.nextLine());
114 if (choice < min || choice > max){ choice = -1;}
115 } catch (Exception e) {
116 choice = -1;
117 }
118 return choice;
119 }
120
121 }
122

src > dao > AuthorDao.java > AuthorDao > getAllAuthors4Book(int)

```java
package dao;

import java.sql.*;
import java.text.Normalizer;
import java.util.*;

import entity.Author;

public class AuthorDao {

    private Connection connection;
    private List<Author> authors;
    private final String GET_ALL_AUTHORS_QUERY = "SELECT authorID, FirstName, MiddleName, LastName, aka FROM Author";
    private final String GET_ALL_P_BOOKS_QUERY = "SELECT A.authorID, A.FirstName, A.MiddleName, A.LastName, A.aka, BA.bookID FROM author AS A INNER JOIN bookauthor AS BA ON A.aut
    private final String ADD_AUTHOR = "INSERT INTO mybooks.author (firstname, middlename, lastname, aka) VALUES(?,?,?,?)";
    private final String GET_NEW_ID = "SELECT authorID FROM author ORDER BY authorID DESC LIMIT 1";

    public AuthorDao() {
        this.connection = DBConnection.getConnection();
    }

    public List<Author> getAllAuthors() throws SQLException{
        ResultSet rs = connection.prepareStatement(GET_ALL_AUTHORS_QUERY).executeQuery();

        authors = new ArrayList<>();
        while (rs.next()){
            authors.add(populateAuthor(rs.getInt(columnIndex: 1), rs.getString(columnIndex: 2), rs.getString(columnIndex: 3), rs.getString(columnIndex: 4),
                rs.getString(columnIndex: 5)));
        }
        return authors;
    }

    public List<Author> getAllAuthors4Book(int bookID) throws SQLException{
        PreparedStatement ps = connection.prepareStatement(GET_ALL_P_BOOKS_QUERY);
        ps.setInt(parameterIndex: 1, bookID);
        ResultSet rs = ps.executeQuery();

        authors = new ArrayList<>();
        while (rs.next()){
            authors.add(populateAuthor(rs.getInt(columnIndex: 1), rs.getString(columnIndex: 2), rs.getString(columnIndex: 3), rs.getString(columnIndex: 4),
                rs.getString(columnIndex: 5)));
        }
        return authors;
    }

    public int addAuthor(String firstName, String middleName, String lastName, String aka) throws SQLException{
        PreparedStatement ps = connection.prepareStatement(ADD_AUTHOR);
        ps.setString(parameterIndex: 1, firstName);
        ps.setString(parameterIndex: 2, middleName);
        ps.setString(parameterIndex: 3, lastName);
        ps.setString(parameterIndex: 4, aka);
        ps.executeUpdate();

        ResultSet rs = connection.prepareStatement(GET_NEW_ID).executeQuery();
        if(rs.next()){
            return rs.getInt(columnIndex: 1);
        }
        return 0;
    }

    private Author populateAuthor (int id, String firstName, String middleName, String lastName, String aka){
        return new Author(id, removeDiacritics(firstName), removeDiacritics(middleName), removeDiacritics(lastName),removeDiacritics(aka));
    }

    private String removeDiacritics(String diacritics){
        if(diacritics == null){
            return null;
        }
        return Normalizer.normalize(diacritics, Normalizer.Form.NFD).replaceAll(regex: "[^\\p{ASCII}]", replacement: "");
    }
}
```

src > dao > BookDao.java > BookDao > getBooksByAuthor(Author)

```java
package dao;

import java.sql.*;
import java.text.Normalizer;
import java.util.ArrayList;
import java.util.List;

import entity.Author;
import entity.Book;
import entity.Language;

public class BookDao {
    private Connection connection;
    private List<Book> books;
    private LanguageDao languageDao = new LanguageDao();
    private AuthorDao authorDao = new AuthorDao();
    private final String GET_ALL_BOOKS = "SELECT * FROM book";
    private final String GET_BOOK_BY_ID = "SELECT * FROM book WHERE bookID = ?";
    private final String GET_BOOKS_BY_AUTHOR = "SELECT * FROM book AS B INNER JOIN bookauthor AS BA ON BA.bookID = B.bookID WHERE BA.authorID = ?";
    private final String INSERT_NEW_BOOK = "INSERT INTO mybooks.book (OriginalBookName, FirstEditionYear,OriginalLanguageID) VALUES (?, ?, ?)";
    private final String INSERT_NEW_BOOKAUTHOR = "INSERT INTO mybooks.bookauthor (BookID, AuthorID) VALUES (?, ?)";
    private final String GET_NEW_BOOKID = "SELECT bookID FROM book ORDER BY bookID DESC LIMIT 1";


    public BookDao() {
        this.connection = DBConnection.getConnection();
    }

    public List<Book> getAllBooks() throws SQLException{
        ResultSet rs = connection.prepareStatement(GET_ALL_BOOKS).executeQuery();

        books = new ArrayList<>();
        while (rs.next()){
            books.add(populateBook(rs.getInt(columnIndex: 1), rs.getString(columnIndex: 2), rs.getInt(columnIndex: 3), rs.getInt(columnIndex: 4)));
        }
        return books;
    }

    public Book getBookByID(int bookID) throws SQLException{
        PreparedStatement ps = connection.prepareStatement(GET_BOOK_BY_ID);
        ps.setInt(parameterIndex: 1, bookID);
        ResultSet rs = ps.executeQuery();

        Book currentBook = null;
        while(rs.next()){
            currentBook = populateBook(rs.getInt(columnIndex: 1), rs.getString(columnIndex: 2), rs.getInt(columnIndex: 3), rs.getInt(columnIndex: 4));
        }
        return currentBook;
    }
```

src > dao > BookDao.java > BookDao > getBooksByAuthor(Author)

```java
    public List<Book> getBooksByAuthor(Author author) throws SQLException{
        PreparedStatement ps = connection.prepareStatement(GET_BOOKS_BY_AUTHOR);
        ps.setInt(parameterIndex: 1, author.getAuthorID());
        ResultSet rs = ps.executeQuery();
        List<Book> books = new ArrayList<>();
        while (rs.next()){
            books.add(populateBook(rs.getInt(columnIndex: 1), rs.getString(columnIndex: 2), rs.getInt(columnIndex: 3), rs.getInt(columnIndex: 4)));
        }
        return books;
    }

    public int createBook(String originalBookName, int yearPublished, int originalLanguageID, List<Integer> authors) throws SQLException{
        PreparedStatement ps = connection.prepareStatement(INSERT_NEW_BOOK);
        ps.setString(parameterIndex: 1, originalBookName);
        ps.setInt(parameterIndex: 2, yearPublished);
        ps.setInt(parameterIndex: 3, originalLanguageID);
        ps.executeUpdate();

        ResultSet rs = connection.prepareStatement(GET_NEW_BOOKID).executeQuery();
        if (rs.next()){
            createBookAuthor(rs.getInt(columnIndex: 1), authors);
            return rs.getInt(columnIndex: 1);
        }
        return 0;
    }

    public void createBookAuthor(int bookID, List<Integer> authors) throws SQLException{
        for (int i : authors){
            PreparedStatement ps = connection.prepareStatement(INSERT_NEW_BOOKAUTHOR);
            ps.setInt(parameterIndex: 1, bookID);
            ps.setInt(parameterIndex: 2, i);
            ps.executeUpdate();
        }

    }

    private Book populateBook (int id, String originalBookName, int yearPublished, int languageID) throws SQLException{
        Language language = languageDao.getLanguageByID(languageID);
        List<Author> authors = authorDao.getAllAuthors4Book(id);

        return new Book(id, removeDiacritics(originalBookName), yearPublished, language, authors);
    }

    private String removeDiacritics(String diacritics){
        if(diacritics == null){
            return null;
        }
        return Normalizer.normalize(diacritics, Normalizer.Form.NFD).replaceAll(regex: "[^\\p{ASCII}]", replacement: "");
    }
```

src > dao > DBConnection.java > DBConnection > getConnection()

```java
package dao;

import java.sql.*;


public class DBConnection {
    private final static String URL = "jdbc:mysql://localhost:3306/mybooks";
    private final static String USERNAME = "root";
    private final static String PASSWORD = "qwer1234!";
    private static Connection connection;
    private static DBConnection instance;

    private DBConnection(Connection connection){
        DBConnection.connection = connection;
    }

    public static Connection getConnection() {
        if (instance == null){
            try {
                    connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
                    instance = new DBConnection(connection);
                    System.out.println(x: "connection successful");
                }
                catch(SQLException e){
                    System.out.println(x: "Access denied");
                    e.printStackTrace();
                }
        }
        return DBConnection.connection;
    }
}

```

src > dao > LanguageDao.java > LanguageDao > addLanguage(String)

```java
package dao;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

import entity.Language;

public class LanguageDao {
    private Connection connection;
    private final String GET_ALL_LANGUAGES = "SELECT * FROM languages";
    private final String GET_LANGUAGE_BY_ID_QUERY = "SELECT * FROM languages WHERE languageID = ?";
    private final String ADD_LANGUAGE = "INSERT INTO mybooks.languages (LanguageDescription) VALUES (?)";
    private final String GET_NEW_LANGUAGEID = "SELECT languageID FROM Languages ORDER BY languageID DESC LIMIT 1";

    public LanguageDao() {
        this.connection = DBConnection.getConnection();
    }

    public List<Language> getAllLanguages() throws SQLException{
        List<Language> languages = new ArrayList<>(
        ResultSet rs = connection.prepareStatement(GET_ALL_LANGUAGES).executeQuery();
        while (rs.next()){
            languages.add(new Language(rs.getInt(columnIndex: 1), rs.getString(columnIndex: 2)));
        }
        return languages;
    }

    public Language getLanguageByID(int languageID) throws SQLException{

        PreparedStatement ps = connection.prepareStatement(GET_LANGUAGE_BY_ID_QUERY);
        ps.setInt(parameterIndex: 1, languageID);
        ResultSet rs = ps.executeQuery();
        Language language = null;
        while (rs.next()){
            language = new Language(rs.getInt(columnIndex: 1), rs.getString(columnIndex: 2));
        }

        return language;
    }
```

String GET_ALL_LANGUAGES

```java

    public int addLanguage(String language) throws SQLException{
        PreparedStatement ps = connection.prepareStatement(ADD_LANGUAGE);
        ps.setString(parameterIndex: 1, language);
        ps.executeUpdate();

        ResultSet rs = connection.prepareStatement(GET_NEW_LANGUAGEID).executeQuery();
        if (rs.next()){
            return rs.getInt(columnIndex: 1);
        }
        return 0;
    }

}
```

src > dao > PublishedBookDao.java > PublishedBookDao > getAllPBooks4Author(Author)

```java
package dao;

import java.sql.*;
import java.text.Normalizer;
import java.util.*;
import java.util.Date;

import entity.Author;
import entity.Book;
import entity.Language;
import entity.PublishedBook;
import entity.Publisher;

public class PublishedBookDao {
    private Connection connection;
    private BookDao bookDao = new BookDao();
    private LanguageDao languageDao = new LanguageDao();


    private final String GET_ALL_P_BOOKS_QUERY = "SELECT * FROM publishedbook";
    private final String GET_P_BOOKS_4_BOOK_QUERY = "SELECT * FROM publishedbook WHERE bookID = ?";
    private final String GET_PUBLISHER_BY_ID = "SELECT * FROM publisher WHERE publisherID = ?";
    private final String GET_FINISHED_DATE = "SELECT DateFinished FROM finishedpublishedbook Where PublishedBookID = ?";
    private final String GET_ALL_PUBLISHERS = "SELECT * FROM publisher";
    private final String INSERT_PBOOK = "INSERT INTO mybooks.publishedbook (BookID, PublishedBookName, YearPublished, PublisherID, LanguageID, Translator)"+
                                        " VALUES (?,?,?,?,?,?)";
    private final String GET_NEW_PBOOKID = "SELECT publishedbookID FROM publishedbook ORDER BY publishedBookID DESC LIMIT 1";
    private final String INSERT_PUBLISHER = "INSERT INTO mybooks.publisher (publisherName) VALUES (?)";
    private final String GET_NEW_PUBLISHERID = "SELECT publisherID FROM publisher ORDER BY publisherID DESC LIMIT 1";
    private final String ADD_FINISHDATE = "INSERT INTO  mybooks.finishedpublishedbook (publishedbookID, dateFinished) VALUES (?, ?)";
    private final String UPDATE_FINISHDATE = "UPDATE mybooks.finishedpublishedbook SET datefinished = ? WHERE publishedbookID = ?";
    private final String DELETE_FINISHEDDATE = "DELETE FROM mybooks.finishedpublishedbook WHERE publishedbookid = ?";
    private final String DELETE_PBOOK = "DELETE FROM mybooks.publishedbook WHERE publishedbookid = ?";
    private final String GET_PBOOKS_BY_BOOKID = "SELECT * FROM publishedbook WHERE bookid = ?";

    public PublishedBookDao() {
        this.connection = DBConnection.getConnection();
    }

    public List<PublishedBook> getAllBooks() throws SQLException{
        ResultSet rs = connection.prepareStatement(GET_ALL_P_BOOKS_QUERY).executeQuery();
        List<PublishedBook> allPBooks = new ArrayList<>();
        while (rs.next()){
            allPBooks.add(populatePBook(rs.getInt(columnIndex: 1), rs.getInt(columnIndex: 2), rs.getString(columnIndex: 3), rs.getInt(columnIndex: 4),
                rs.getInt(columnIndex: 5), rs.getInt(columnIndex: 6), rs.getString(columnIndex: 7)));
        }
        return allPBooks;
    }
```

src > dao > PublishedBookDao.java > PublishedBookDao > getAllPBooks4Author(Author)

```
48      }
49
50      public List<PublishedBook> getAllPBooks4Author(Author author) throws SQLException{
51          List<Book> books = bookDao.getBooksByAuthor(author);
52          List<PublishedBook> allPBooks = new ArrayList<>();
53          for (Book book: books){
54
55              PreparedStatement ps = connection.prepareStatement(GET_P_BOOKS_4_BOOK_QUERY);
56              ps.setInt(parameterIndex: 1, book.getBookID());
57              ResultSet rs = ps.executeQuery();
58              while (rs.next()){
59                  allPBooks.add(populatePBook(rs.getInt(columnIndex: 1), rs.getInt(columnIndex: 2), rs.getString(columnIndex: 3), rs.getInt(columnIndex: 4),
60                      rs.getInt(columnIndex: 5), rs.getInt(columnIndex: 6), rs.getString(columnIndex: 7)));
61              }
62          }
63          return allPBooks;
64      }
65
66      public int addPBook(int bookID, String title, int year, int pulisherID, int languageID, String translator) throws SQLException{
67          PreparedStatement ps = connection.prepareStatement(INSERT_PBOOK);
68          ps.setInt(parameterIndex: 1, bookID);
69          ps.setString(parameterIndex: 2, title);
70          ps.setInt(parameterIndex: 3, year);
71          ps.setInt(parameterIndex: 4, pulisherID);
72          ps.setInt(parameterIndex: 5, languageID);
73          ps.setString(parameterIndex: 6, translator);
74          ps.executeUpdate();
75
76          ResultSet rs = connection.prepareStatement(GET_NEW_PBOOKID).executeQuery();
77          if (rs.next()){
78              return rs.getInt(columnIndex: 1);
79          }
80          return 0;
81      }
82
83      public int addPublisher(String publisherName) throws SQLException{
84          PreparedStatement ps = connection.prepareStatement(INSERT_PUBLISHER);
85          ps.setString(parameterIndex: 1, publisherName);
86          ps.executeUpdate();
87
88          ResultSet rs = connection.prepareStatement(GET_NEW_PUBLISHERID).executeQuery();
89          if(rs.next()){
90              return rs.getInt(columnIndex: 1);
91          }
92          return 0;
93      }
```

src > dao > PublishedBookDao.java > PublishedBookDao > getAllPBooks4Author(Author)

```
    }

    public void addFinishedDate (int pBookID, Date finishedDate, boolean update) throws SQLException{
        java.sql.Date sqlDate = new java.sql.Date(finishedDate.getTime());
        if (!update){
            PreparedStatement ps = connection.prepareStatement(ADD_FINISHDATE);
            ps.setInt(parameterIndex: 1, pBookID);
            ps.setDate(parameterIndex: 2, sqlDate);
            ps.executeUpdate();
        } else{
            PreparedStatement ps = connection.prepareStatement(UPDATE_FINISHDATE);
            ps.setDate(parameterIndex: 1, sqlDate);
            ps.setInt(parameterIndex: 2, pBookID);
            ps.executeUpdate();
        }
    }

    public void deletePBook(int pBookID) throws SQLException{
        PreparedStatement psfin = connection.prepareStatement(DELETE_FINISHEDDATE);
        psfin.setInt(parameterIndex: 1, pBookID);
        psfin.executeUpdate();

        PreparedStatement psp = connection.prepareStatement(DELETE_PBOOK);
        psp.setInt(parameterIndex: 1, pBookID);
        psp.executeUpdate();
    }

    public List<PublishedBook> getAllPBooks4BookID(int bookID) throws SQLException{
        List<PublishedBook> pBooks = new ArrayList<>();
        PreparedStatement ps = connection.prepareStatement(GET_PBOOKS_BY_BOOKID);
        ps.setInt(parameterIndex: 1, bookID);
        ResultSet rs = ps.executeQuery();
        while (rs.next()){
            pBooks.add(populatePBook(rs.getInt(columnIndex: 1), rs.getInt(columnIndex: 2), rs.getString(columnIndex: 3), rs.getInt(columnIndex: 4), rs.getInt(columnIndex: 5),
                    rs.getInt(columnIndex: 6), rs.getString(columnIndex: 7)));
        }
        return pBooks;
    }

    private PublishedBook populatePBook(int id, int bookID, String publishedBookName, int yearPublished, int publisherID, int languageID, String translator) throws SQLExcepti
        Book book = bookDao.getBookByID(bookID);
        Publisher publisher = getPublisherByID(publisherID);
        Language language = languageDao.getLanguageByID(languageID);
        Date finished = finished(id);

        return new PublishedBook(id, book, removeDiacritics(publishedBookName), yearPublished, publisher, language, removeDiacritics(translator),finished);
    }

    private Publisher getPublisherByID(int id) throws SQLException{
```

src > dao > PublishedBookDao.java > PublishedBookDao > getAllPBooks4Author(Author)

```
        return new PublishedBook(id, book, removeDiacritics(publishedBookName), yearPublished, publisher, language, removeDiacritics(translator),finished);
    }

    private Publisher getPublisherByID(int id) throws SQLException{
        PreparedStatement ps = connection.prepareStatement(GET_PUBLISHER_BY_ID);
        ps.setInt(parameterIndex: 1, id);
        ResultSet rs = ps.executeQuery();

        Publisher publisher = null;
        while (rs.next()){
            publisher = new Publisher(rs.getInt(columnIndex: 1), rs.getString(columnIndex: 2));
        }

        return publisher;
    }

    public List<Publisher> getAllPublishers() throws SQLException{
        ResultSet rs = connection.prepareStatement(GET_ALL_PUBLISHERS).executeQuery();
        List<Publisher> publishers = new ArrayList<>();
        while(rs.next()){
            publishers.add(new Publisher(rs.getInt(columnIndex: 1),rs.getString(columnIndex: 2)));
        }
        return publishers;
    }
    private Date finished(int pBookID) throws SQLException{
        Date date = null;
        PreparedStatement ps = connection.prepareStatement(GET_FINISHED_DATE);
        ps.setInt(parameterIndex: 1, pBookID);
        ResultSet rs = ps.executeQuery();
        if (rs.next()){ date = rs.getDate(columnIndex: 1);}
        return date;
    }

    private String removeDiacritics(String diacritics){
        if(diacritics == null){
            return null;
        }
        return Normalizer.normalize(diacritics, Normalizer.Form.NFD).replaceAll(regex: "[^\\p{ASCII}]", replacement: "");
    }
}
```

src > entity > Author.java > Author

```java
package entity;

public class Author {
    private int authorID;
    private String firstName;
    private String middleName;
    private String lastName;
    private String aka;
    private boolean hasAka;

    public Author(int authorID, String firstName, String middleName, String lastName, String aka) {
        this.authorID = authorID;
        this.firstName = firstName;
        this.middleName = middleName;
        this.lastName = lastName;
        this.aka = aka;
        this.hasAka = !(aka==null);
    }

    public void printDetails(){
        System.out.print( authorID + ". " + firstName);
        if (middleName != null){System.out.print(" " + middleName);}
        if (lastName != null){System.out.print(" " + lastName);}
        if (aka != null){
            if(!aka.equals(anObject: "")){System.out.print(" AKA: " + aka);}
        }
        System.out.println();
    }

    public String getName(){
        String name = firstName;
        if (lastName != null){name += " " + lastName;}
        return name;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getMiddleName() {
        return middleName;
    }
```

src > entity > Author.java > Author

```
38      }
39
40      public void setFirstName(String firstName) {
41          this.firstName = firstName;
42      }
43
44      public String getMiddleName() {
45          return middleName;
46      }
47
48      public void setMiddleName(String middleName) {
49          this.middleName = middleName;
50      }
51
52      public String getLastName() {
53          return lastName;
54      }
55
56      public void setLastName(String lastName) {
57          this.lastName = lastName;
58      }
59
60      public String getAka() {
61          return aka;
62      }
63
64      public void setAka(String aka) {
65          this.aka = aka;
66      }
67
68      public boolean hasAka() {
69          return hasAka;
70      }
71
72      public void setHasAka(boolean hasAka) {
73          this.hasAka = hasAka;
74      }
75
76      public int getAuthorID() {
77          return authorID;
78      }
79
80  }
81
```

src > entity > Book.java > Book > printDetails()

```java
package entity;

import java.util.List;

public class Book {
    private int bookID;
    private String originalBookName;
    private int firstPublishedYear;
    private Language originalLanguage;
    private List<Author> authors;

    public Book (String originalBookName, int firstPublishedYear, Language originalLanguage, List<Author> authors){
        this.originalBookName = originalBookName;
        this.firstPublishedYear = firstPublishedYear;
        this.originalLanguage = originalLanguage;
        this.authors = authors;
    }

    public Book (int bookID, String originalBookName, int firstPublishedYear, Language originalLanguage, List<Author> authors){
        this.bookID = bookID;
        this.originalBookName = originalBookName;
        this.firstPublishedYear = firstPublishedYear;
        this.originalLanguage = originalLanguage;
        this.authors = authors;
    }

    public void printDetails(){
        System.out.print(bookID + ". " + originalBookName + " written by ");
        for (int i = 0; i < authors.size(); i++){
            if (i < authors.size() - 1){
                System.out.print(authors.get(i).getName() + " and ");
            } else {
                System.out.print(authors.get(i).getName());
            }
        }
        System.out.println();
    }

    public int getBookID() {
        return bookID;
    }

    public String getOriginalBookName() {
        return originalBookName;
    }
```

```
        }
        System.out.println();
    }

    public int getBookID() {
        return bookID;
    }

    public String getOriginalBookName() {
        return originalBookName;
    }

    public int getFirstPublishedYear() {
        return firstPublishedYear;
    }

    public Language getOriginalLanguage() {
        return originalLanguage;
    }

    public List<Author> getAuthors() {
        return authors;
    }

    public void setOriginalBookName(String originalBookName) {
        this.originalBookName = originalBookName;
    }

    public void setFirstPublishedYear(int firstPublishedYear) {
        this.firstPublishedYear = firstPublishedYear;
    }

    public void setOriginalLanguage(Language originalLanguage) {
        this.originalLanguage = originalLanguage;
    }

    public void setAuthors(List<Author> authors) {
        this.authors = authors;
    }

}
```

src > entity > Language.java > Language

```java
package entity;

public class Language {
    private int languageID;
    private String languageDescription;

    public Language(String languageDescription) {
        this.languageDescription = languageDescription;
    }

        public Language(int languageID, String languageDescription) {
        this.languageID = languageID;
        this.languageDescription = languageDescription;
    }

    public void printDetails(){
        System.out.println(languageID + ". " + languageDescription);
    }

    public int getLanguageID() {
        return languageID;
    }

    public String getLanguageDescription() {
        return languageDescription;
    }

    public void setLanguageDescription(String languageDescription) {
        this.languageDescription = languageDescription;
    }

}

```

src > entity > PublishedBook.java > PublishedBook

```java
package entity;

import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.List;

public class PublishedBook {
    private int publishedBookID;
    private Book book;
    private String publishedBookName;
    private int yearPublished;
    private Publisher publisher;
    private Language language;
    private String translator;
    private boolean finished = false;
    private Date dateFinished;


    public PublishedBook(int publishedBookID, Book book, String publishedBookName, int yearPublished,
            Publisher publisher, Language language, String translator, Date dateFinished) {
        this.publishedBookID = publishedBookID;
        this.book = book;
        this.publishedBookName = publishedBookName;
        this.yearPublished = yearPublished;
        this.publisher = publisher;
        this.language = language;
        this.translator = translator;
        this.dateFinished = dateFinished;
        if (dateFinished != null){this.finished = true;}
    }

    public void printDetails(){
        DateFormat df = new SimpleDateFormat(pattern: "dd MMM yyyy");
        List<Author> authors = book.getAuthors();
        System.out.print(publishedBookName + " written by ");
        for (int i = 0; i < authors.size(); i++){
            if (i < authors.size() - 1){
                System.out.print(authors.get(i).getName() + " and ");
            } else {
                System.out.print(authors.get(i).getName());
            }
        }
        if (dateFinished != null){ System.out.print(" Finished book on: " + df.format(dateFinished));}
    }

```

```java
public int getPublishedBookID() {
    return publishedBookID;
}

public Book getBook() {
    return book;
}

public String getPublishedBookName() {
    return publishedBookName;
}

public int getYearPublished() {
    return yearPublished;
}

public Publisher getPublisher() {
    return publisher;
}

public Language getLanguage() {
    return language;
}

public String getTranslator() {
    return translator;
}

public boolean isFinished() {
    return finished;
}

public Date getDateFinished() {
    return dateFinished;
}

public void setBook(Book book) {
    this.book = book;
}

public void setPublishedBookName(String publishedBookName) {
    this.publishedBookName = publishedBookName;
}

public void setYearPublished(int yearPublished) {
    this.yearPublished = yearPublished;
}
```

src > entity > PublishedBook.java > PublishedBook

```
77      }
78
79      public Date getDateFinished() {
80          return dateFinished;
81      }
82
83      public void setBook(Book book) {
84          this.book = book;
85      }
86
87      public void setPublishedBookName(String publishedBookName) {
88          this.publishedBookName = publishedBookName;
89      }
90
91      public void setYearPublished(int yearPublished) {
92          this.yearPublished = yearPublished;
93      }
94
95      public void setPublisher(Publisher publisher) {
96          this.publisher = publisher;
97      }
98
99      public void setLanguage(Language language) {
100         this.language = language;
101     }
102
103     public void setTranslator(String translator) {
104         this.translator = translator;
105     }
106
107     public void setFinished(boolean finished) {
108         this.finished = finished;
109     }
110
111     public void setDateFinished(Date dateFinished) {
112         this.dateFinished = dateFinished;
113     }
114
115 }
116
```

src > entity > Publisher.java > Publisher > printDetails()

```java
package entity;

public class Publisher {
    private int publisherID;
    private String name;

    public Publisher(String name){
        this.name = name;
    }

    public Publisher(int publisherID, String name) {
        this.publisherID = publisherID;
        this.name = name;
    }

    public void printDetails(){
        System.out.println(publisherID + ". " + name);
    }
    public int getPublisherID() {
        return publisherID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

Screenshots of Running Application:

```
p=transport=dt_socket,server=n,suspend=y,address=localhost:64761' '
connection successful

Please select one of the following options and then hit enter:
1. See all books in your library.
2. See all authors in your library.
3. See how many versions of a specific book you have.
4. Add a new book.
5. Set a book to finished.
6. Remove a book from your library.
7. Exit.
```

```
6. Remove a book from your library.
7. Exit.

1
1. Dasenka cili zivot stenete written by Karel Capek Finished book on: 30 Aug 1990
2. Dasja written by Karel Capek
3. Dashenka Or, The Life of a Puppy written by Karel Capek
4. Neverwhere written by Neil Gaiman Finished book on: 04 Mar 2000
5. Nikdykde written by Neil Gaiman Finished book on: 07 May 2000
6. Niemandsland written by Neil Gaiman
7. Color of Magic written by Terry Pratchett
8. Barva kouzel written by Terry Pratchett
9. De kleur van toverij written by Terry Pratchett
10. Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch written by Terry Pratchett and Neil Gaiman
11. Dobra znameni written by Terry Pratchett and Neil Gaiman
12. Hoge Omens written by Terry Pratchett and Neil Gaiman
13. De donkere kamer van Damokles written by Willem Hermans Finished book on: 21 Sep 2002
14. The Darkroom of Damocles written by Willem Hermans
15. Temna komora Damoklova written by Willem Hermans

Please select one of the following options and then hit enter:
1. See all books in your library.
2. See all authors in your library.
3. See how many versions of a specific book you have.
4. Add a new book.
5. Set a book to finished.
6. Remove a book from your library.
7. Exit.
```

```
1. See all books in your library.
2. See all authors in your library.
3. See how many versions of a specific book you have.
4. Add a new book.
5. Set a book to finished.
6. Remove a book from your library.
7. Exit.

2
1. Willem Frederik Hermans AKA: WF Hermans
2. Karel Capek
3. Terry Pratchett AKA: Sir Terence David John Pratchett
4. Neil Richard Gaiman AKA: Neil Richard MacKinnon Gaiman

Please choose one of the following options and hit enter:
1. To see all the books in your library by a specific author
2. Show all authors again
3. Go back
1

Please select the id of an author listed above and hit enter
2
1. Dasenka cili zivot stenete written by Karel Capek Finished book on: 30 Aug 1990
2. Dasja written by Karel Capek
3. Dashenka Or, The Life of a Puppy written by Karel Capek

Please choose one of the following options and hit enter:
1. To see all the books in your library by a specific author
2. Show all authors again
3. Go back
3

Please select one of the following options and then hit enter:
1. See all books in your library.
2. See all authors in your library.
3. See how many versions of a specific book you have.
4. Add a new book.
5. Set a book to finished.
6. Remove a book from your library.
7. Exit.
```

```
3. See how many versions of a specific book you have.
4. Add a new book.
5. Set a book to finished.
6. Remove a book from your library.
7. Exit.

3
Which book do you want to see all the versions of?
0. go back to the main menu
1. Dasenka cili zivot stenete written by Karel Capek
2. Neverwhere written by Neil Gaiman
3. The Colour of Magic written by Terry Pratchett
4. Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch written by Terry Pratchett and Neil Gaiman
5. De donkere kamer van Damokles written by Willem Hermans
4
1. Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch written by Terry Pratchett and Neil Gaiman
2. Dobra znameni written by Terry Pratchett and Neil Gaiman
3. Hoge Omens written by Terry Pratchett and Neil Gaiman

Please select one of the following options and then hit enter:
1. See all books in your library.
2. See all authors in your library.
3. See how many versions of a specific book you have.
4. Add a new book.
5. Set a book to finished.
6. Remove a book from your library.
7. Exit.
```

```
4. Add a new book.
5. Set a book to finished.
6. Remove a book from your library.
7. Exit.

4
Please make a selection from the following menu and hit enter:
1. I want to add a book
2. I want to go back to the main menu
1
We're going to start with information about the original edition of this book
This information is usually foind on the first couple of pages of a book
Please select 0 if your book is not one of these books or select the book's ID to add a new version
0. This is a completely new book
1. Dasenka cili zivot stenete written by Karel Capek
2. Neverwhere written by Neil Gaiman
3. The Colour of Magic written by Terry Pratchett
4. Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch written by Terry Pratchett and Neil Gaiman
5. De donkere kamer van Damokles written by Willem Hermans
0
Please input the original title of the book when it was first published
the original title might be in a different language if the book has been translated
please hit enter when you are done
Pippi Dlouha Puncocha
Please enter the year when the book was first published
1999
What language was the first edition printed in? Please make a selection and hit enter
0. My language isn't on the list
1. English
2. Czech
3. Dutch
0
Please input the language you'd like to input and hit enter
Slovak
Please input who wrote your book
If your book has multiple authors please input one at a time
0. My author is not on the list
1. Willem Frederik Hermans AKA: WF Hermans
2. Karel Capek
3. Terry Pratchett AKA: Sir Terence David John Pratchett
4. Neil Richard Gaiman AKA: Neil Richard MacKinnon Gaiman
5. I am done adding authors
0
```

```
0
please input the author's first name and hit enter
Astrig Lindgren
please input the author's middle name, if they have none just hit enter

please input the author's last name, if they have none just hit enter

Some authors use a nom de plume if the author also goes by a different name please put that in
if the author does not have another name that they go by please just hit enter
Astrid Lindgrenova
[5]
Please input who wrote your book
If your book has multiple authors please input one at a time
0. My author is not on the list
1. Willem Frederik Hermans AKA: WF Hermans
2. Karel Capek
3. Terry Pratchett AKA: Sir Terence David John Pratchett
4. Neil Richard Gaiman AKA: Neil Richard MacKinnon Gaiman
5. Astrig Lindgren   AKA: Astrid Lindgrenova
6. I am done adding authors
6

Now we need information about this specific copy:
Please input the title of this book and hit enter
Pippi Langstumpf
Please input the year this copy was published and hit enter
1988
Please make a selection of the book's publisher and hit enter
0. My publisher is not on this list
1. Albatros
2. Voetnoot, Uitgeverij
3. James Press
4. Talpress
5. De Boekerij
6. Haper Collins Publishers
7. Polaris
8. Uitgeverij Luitingh-Sijthoff B.V.
9. Host
10. The Overlook Press
11. Uitgeverij G.A. van Oorschot B.V.
1
What language is your book in? Please make a selection and hit enter
0. My language isn't on the list
1. English
2. Czech
3. Dutch
4. Slovak
0
```

```
Please input the language you'd like to input and hit enter
Swedish
Was this book translated? Please make input your choice and hit enter
1. Yes, this book was originally written in another language.
2. No, this book is in it's original language
1
Please input the translator's name in and hit enter.
Milada Horackova
SUCCESS your book has been added
Please make a selection from the following menu and hit enter:
1. I want to add a book
2. I want to go back to the main menu
```

```
5. Set a book to finished.
6. Remove a book from your library.
7. Exit.

5
Please select which book you would like to set to finished
0. go back to the main menu
1. Dasenka cili zivot stenete written by Karel Capek Finished book on: 30 Aug 1990
2. Dasja written by Karel Capek
3. Dashenka Or, The Life of a Puppy written by Karel Capek
4. Neverwhere written by Neil Gaiman Finished book on: 04 Mar 2000
5. Nikdykde written by Neil Gaiman Finished book on: 07 May 2000
6. Niemandsland written by Neil Gaiman
7. Color of Magic written by Terry Pratchett
8. Barva kouzel written by Terry Pratchett
9. De kleur van toverij written by Terry Pratchett
10. Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch written by Terry Pratchett and Neil Gaiman
11. Dobra znameni written by Terry Pratchett and Neil Gaiman
12. Hoge Omens written by Terry Pratchett and Neil Gaiman
13. De donkere kamer van Damokles written by Willem Hermans Finished book on: 21 Sep 2002
14. The Darkroom of Damocles written by Willem Hermans
15. Temna komora Damoklova written by Willem Hermans
16. Pippi Langstumpf written by Astrig Lindgren
16
Please input the date that you finished the book in YYYY-MM-DD format
2asdf
Please input the date that you finished the book in YYYY-MM-DD format
2022-05-06
1. Dasenka cili zivot stenete written by Karel Capek Finished book on: 30 Aug 1990
2. Dasja written by Karel Capek
3. Dashenka Or, The Life of a Puppy written by Karel Capek
4. Neverwhere written by Neil Gaiman Finished book on: 04 Mar 2000
5. Nikdykde written by Neil Gaiman Finished book on: 07 May 2000
6. Niemandsland written by Neil Gaiman
7. Color of Magic written by Terry Pratchett
8. Barva kouzel written by Terry Pratchett
9. De kleur van toverij written by Terry Pratchett
10. Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch written by Terry Pratchett and Neil Gaiman
11. Dobra znameni written by Terry Pratchett and Neil Gaiman
12. Hoge Omens written by Terry Pratchett and Neil Gaiman
13. De donkere kamer van Damokles written by Willem Hermans Finished book on: 21 Sep 2002
14. The Darkroom of Damocles written by Willem Hermans
15. Temna komora Damoklova written by Willem Hermans
16. Pippi Langstumpf written by Astrig Lindgren  Finished book on: 06 May 2022
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

6. Remove a book from your library.
7. Exit.

6
Please select which book you would like to delete and hit enter
0. I don't want to delete any books, take me back to the main menu
1. Dasenka cili zivot stenete written by Karel Capek Finished book on: 30 Aug 1990
2. Dasja written by Karel Capek
3. Dashenka Or, The Life of a Puppy written by Karel Capek
4. Neverwhere written by Neil Gaiman Finished book on: 04 Mar 2000
5. Nikdykde written by Neil Gaiman Finished book on: 07 May 2000
6. Niemandsland written by Neil Gaiman
7. Color of Magic written by Terry Pratchett
8. Barva kouzel written by Terry Pratchett
9. De kleur van toverij written by Terry Pratchett
10. Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch written by Terry Pratchett and Neil Gaiman
11. Dobra znameni written by Terry Pratchett and Neil Gaiman
12. Hoge Omens written by Terry Pratchett and Neil Gaiman
13. De donkere kamer van Damokles written by Willem Hermans Finished book on: 21 Sep 2002
14. The Darkroom of Damocles written by Willem Hermans
15. Temna komora Damoklova written by Willem Hermans
16. Pippi Langstumpf written by Astrig Lindgren  Finished book on: 06 May 2022
12
are you sure you want to delete this book?
Hoge Omens written by Terry Pratchett and Neil Gaiman
1. yes, delete it already!
2. no, let's go back to safety!
1
1. Dasenka cili zivot stenete written by Karel Capek Finished book on: 30 Aug 1990
2. Dasja written by Karel Capek
3. Dashenka Or, The Life of a Puppy written by Karel Capek
4. Neverwhere written by Neil Gaiman Finished book on: 04 Mar 2000
5. Nikdykde written by Neil Gaiman Finished book on: 07 May 2000
6. Niemandsland written by Neil Gaiman
7. Color of Magic written by Terry Pratchett
8. Barva kouzel written by Terry Pratchett
9. De kleur van toverij written by Terry Pratchett
10. Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch written by Terry Pratchett and Neil Gaiman
11. Dobra znameni written by Terry Pratchett and Neil Gaiman
12. De donkere kamer van Damokles written by Willem Hermans Finished book on: 21 Sep 2002
13. The Darkroom of Damocles written by Willem Hermans
14. Temna komora Damoklova written by Willem Hermans
15. Pippi Langstumpf written by Astrig Lindgren  Finished book on: 06 May 2022

Please select one of the following options and then hit enter:
1. See all books in your library.
```

```
Please select one of the following options and then hit enter:
1. See all books in your library.
2. See all authors in your library.
3. See how many versions of a specific book you have.
4. Add a new book.
5. Set a book to finished.
6. Remove a book from your library.
7. Exit.

7
PS C:\Users\fabri\Desktop\Promineo\week10\PromineoWeek10\Week10>
```

URL to GitHub Repository:

https://github.com/MerelOhler/PromineoWeek10/