**Program Fundamentals**

**CA3 Individual Assignment**

**Jan 2023Semester**

**Name: Jeslyn Ho Ka Yan:**

**ID: 10241485**

**Date: 2/6/2023**

**Table of Content**

1 Introduction to Project 3

2 UML Design Class Diagrams (5) 4

3 Explanations of Programs 6

3.1 Book Class 6

3.2 User Account Class 6

3.3 Rental Class 6

3.4 Admin Account 6

3.5 Management System 6

3.5.1 Application Menu Option 7

4 Reflection 11

4.1 What I've learned 11

4.2 Unresolved 12

4.3 Unachievable Requirements 13

5 Appendix A: Java code 14

5.1 Book.java 14

5.2 UserAccount.java 15

5.3 Rental.java 16

5.4 AdminAccount.java 18

5.5 ManagementSystem.java 19

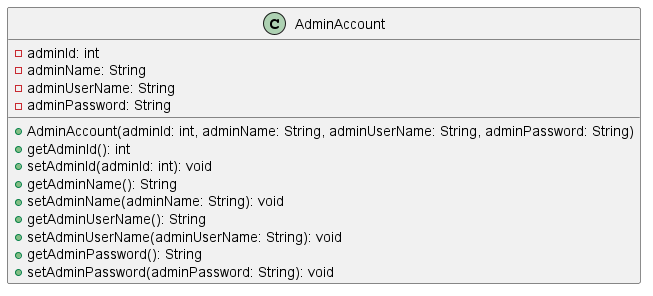
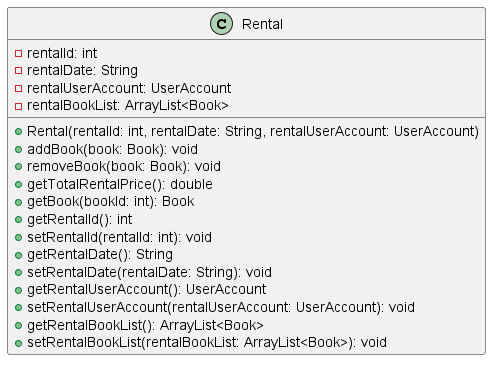
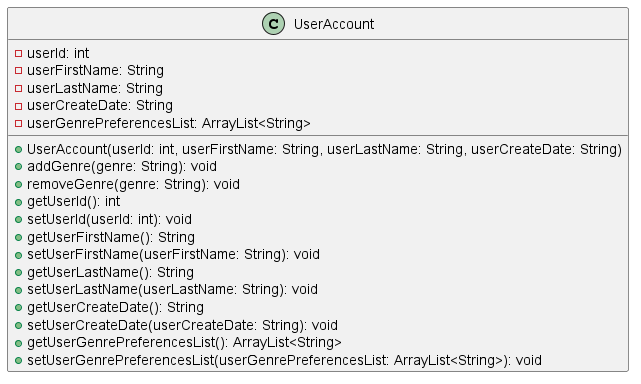
# Introduction to Project

I am a developer from Aethyr IT which is a full-service solution provider that enables businesses to explore the different advantages technology could do for them. We value our customers needs and budgets and strive to perfect a technology curated just for them. I was assigned to develop a simplified “Bookkeep Rental Management” specifically to aid bookkeepers in accessing and renting out books to clients efficiently.

I am required to create 5 Java Classes.

* 1. a. Book
  2. b. UserAccount
  3. c. Rental
  4. d. AdminAccount
  5. e. ManagementSystem (Main Class)

# UML Design Class Diagrams (5)



|  |
| --- |
| Book |
| -bookId: int  -bookName: String  -bookDescription: String  -bookGenre: String  -bookRentalPrice: double |
| +<<constructor>> Book (bookId: int, bookName: String, bookDescription: String, bookGenre: String, bookRentalPrice: double)  +setBookId(bookId: int)  +getBookId(): int  +setBookName(bookName:String)  +getBookName(): String  +setBookDescription(bookDescription:String)  +getBookDescription (): String  +setBookGenre (bookGenre:String)  +getBookGenre (): String  +setBookRentalPrice (bookRentalPrice:String)  +getBookRentalPrice (): double |

|  |
| --- |
| User Account |
| -userId: int  -userFirstName: String  -userLastName: String  -userCreateDate: String  -userGenrePreferencesList: ArrayList |
| +<<constructor>> User Account (userId: int, userFirstName: String, userLastName: String, userCreateDate: String, rentalBookList: ArrayList)  +setUserId (userId: int)  +getUserId (): int  +setUserFirstName (userFirstName:String)  +getUserFirstName (): String  +setUserLastName (userLastName String)  +getUserLastName (): String  +setUserCreateDate (UserCreateDate:String)  +getUserCreateDate (): String  +setUserGenrePreferencesList (UserGenrePreferencesList: ArrayList)  +getUserGenrePreferencesList (): ArrayList |

|  |
| --- |
| Rental |
| -rentalId: int  -rentalDate: String  -rentalUserAccount: UserAccount  -rentalBookList: ArrayList |
| +<<constructor>> Rental (rentalId: int, rentalDate: String, rentalUserAccount: UserAccount, rentalBookList: ArrayList)  +setRentalId (rentalId: int)  +getRentalId (): int  +setRentalDate (rentalDate: String)  +getRentalDate (): String  +setRentalUserAccount: UserAccount  +getRentalUserAccount(): User Account  +setRentalBookList (rentalBookList: ArrayList)  +getRentalBookList (): ArrayList |

|  |
| --- |
| Admin Account |
| -adminId: int  -adminName: String  -adminUserName: String  -adminPassword: String |
| +<<constructor>> Admin Account (adminId: int, adminName: String, adminUserName: String, adminPassword: String)  +setAdminId (adminId: int)  +getAdminId (): int  +setRentalDate (rentalDate: String)  +getRentalDate (): String  +setRentalDate (rentalDate: String)  +getRentalDate (): String  +setRentalDate (rentalDate: String)  +getRentalDate (): String  +setRentalDate (rentalDate: String)  +getRentalDate (): String |

# Explanations of Programs

## Book Class



## User Account Class



## Rental Class



## Admin Account

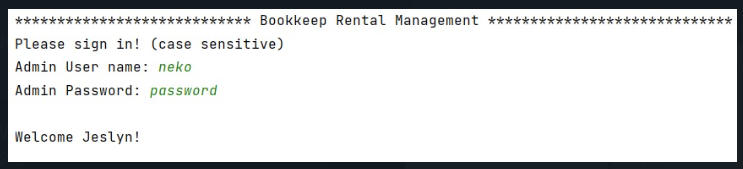


## Management System

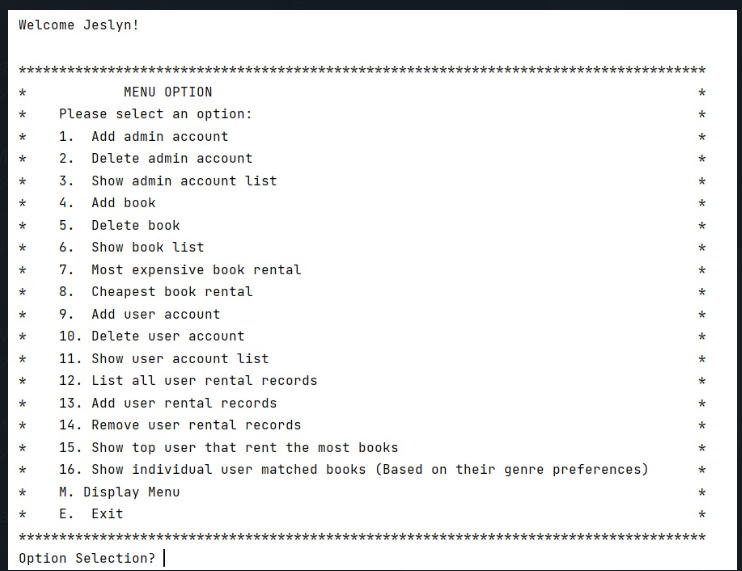


### Application Menu Option

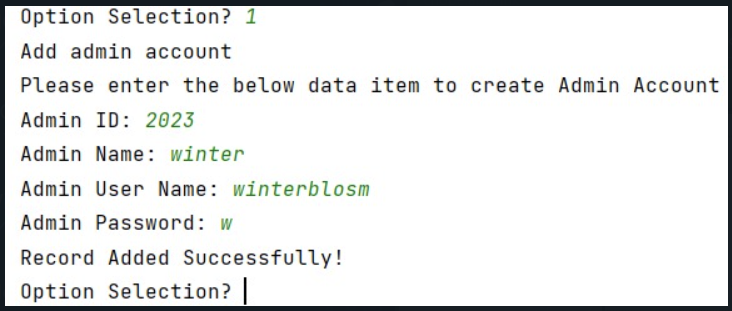
**Firstly, Admin User are required to sign in using their own Username and Password.**



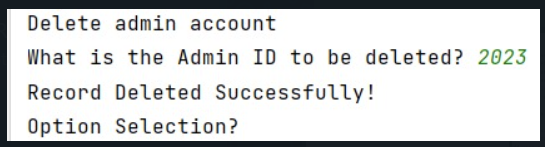
**After Logging in a Menu is then Display. There are 16 option excluding M and E, which is Display menu and Exit the program.**



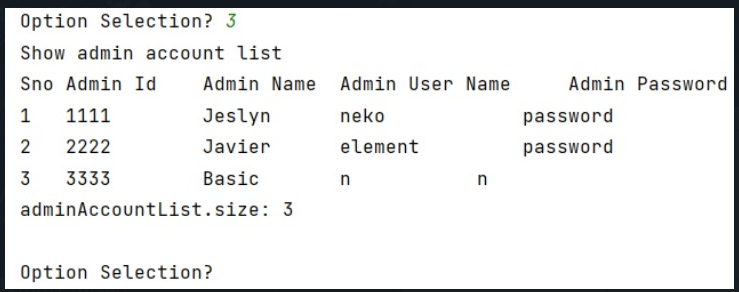
1. **Option 1: Creating a new Admin Account, requires admin to input new admin id, admin name, admin username and password.**



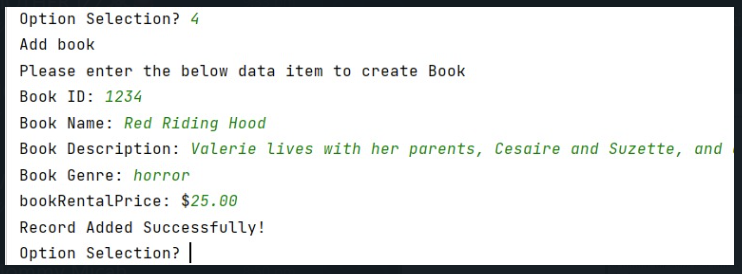
1. **Option 2: Delete Admin Account, admin are required to input Admin ID to delete it.**



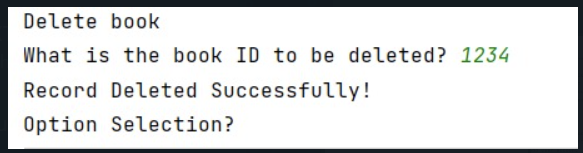
1. **Option 3: Show admin Account List**



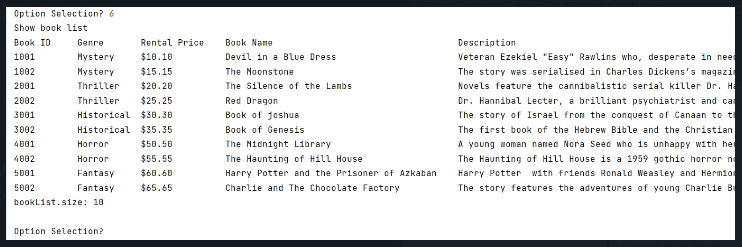
1. **Option 4: Add a new book, admin have to input a new Book id, book name, description of the book, genre and the book rental price.**



1. **Option 5: Delete book by inputting the Book Id**



1. **Option 6: Display Book List**



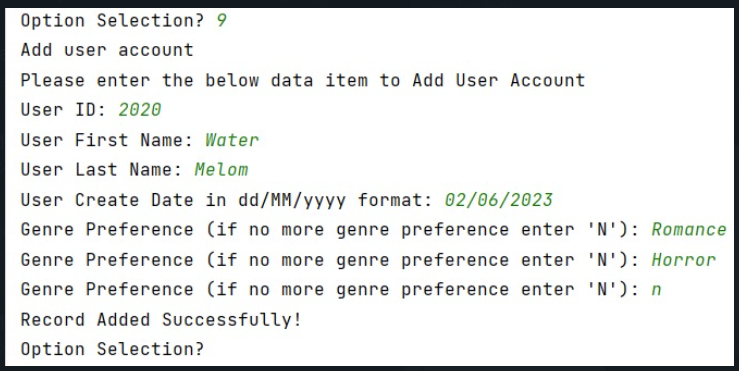
1. **Option 7: Display the Most Expensive Book Rental**



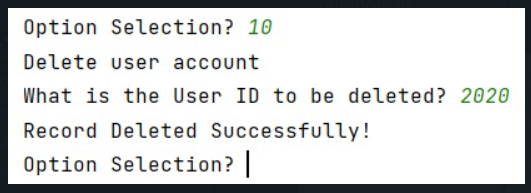
1. **Option 8: Display the Cheapest Book Rental**



1. **Option 9: Add User Account: admin is required to input the user id, user first and last name, the date this user id is created and the user preference.**



1. **Option 10: Delete User: admin have to input the user ID.**



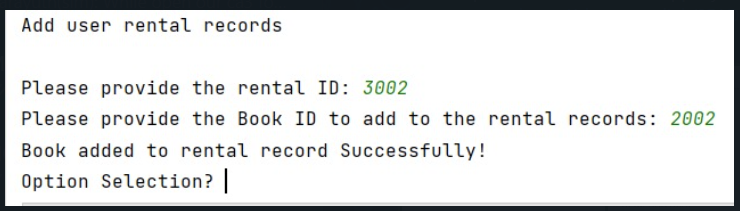
1. **Option 11 Display the User account list.**



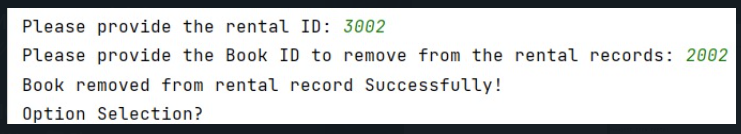
1. **Option 12: Display Rental Records**



1. **Option 13: Add User Rental Records: admin have to provide a rental ID and Book ID to add into the rental records.**



1. **Option 14: Remove Rental Record: admin have to provide a rental ID and Book ID to remove from the rental records.**



# Reflection

## What I've learned

1. A good IDE can really help to code much faster. I explorer using eclipse to write the program beside using IntelliJ.
2. This assignment provides an opportunity for me to learn, practice and strength my java programming and debugging skill.

I realised that when the program is not working as expected, it is a good practice to print out the object to examine the output of each step.

1. It also teaches me how to do refactoring of codes to make it more reusable and readable. After writing the basic skeleton of the program, I realised that there are many replicated codes.

Such as search the list for the object. Hence I wrote a common function to search for the list so that it can be easily called by other functions.

1. It is also important to design the overall program flow before starting on the coding.

Having an overall structure of the program enables me to know the interdependency of the functions

E.g. It is more efficient to code the function for show book list first so that I can use this function to check the result of my Add book and Delete book functions.

1. It is also important to consider user experience during the design of the program.

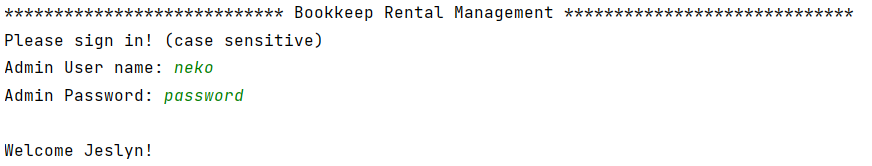
E.g. To delete book, it should also show book list first otherwise user would not know the record id and book id record to remove. Such requirement is not listed in the assignment.

## Unresolved

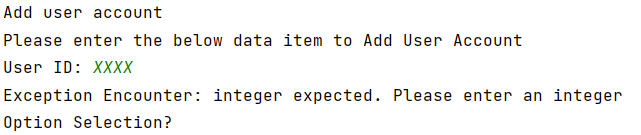
All requirements are fulfilled. However, there can be some future enhancement for consideration.

**Future Enhancement:**

1. Password should be masked and impose restrictions such as must contain 8 characters with combination of upper and lower case to make the authentication secure.



1. Once error exception encounter, system go back to option selection instead of allowing user to re-enter for some functions.



1. When the program start, it will load the Arraylist of adminAccountList, userAccountList, bookList and rentalList from the respective text file. When the program end, it should save by overwrite the text files. Due to insufficient time, I will leave this feature for future enhancement.
2. All the record IDs e.g. user ID, book ID, rental ID should be auto-generated so that user does not need to keep track of the unused numbers.

## Unachievable Requirements

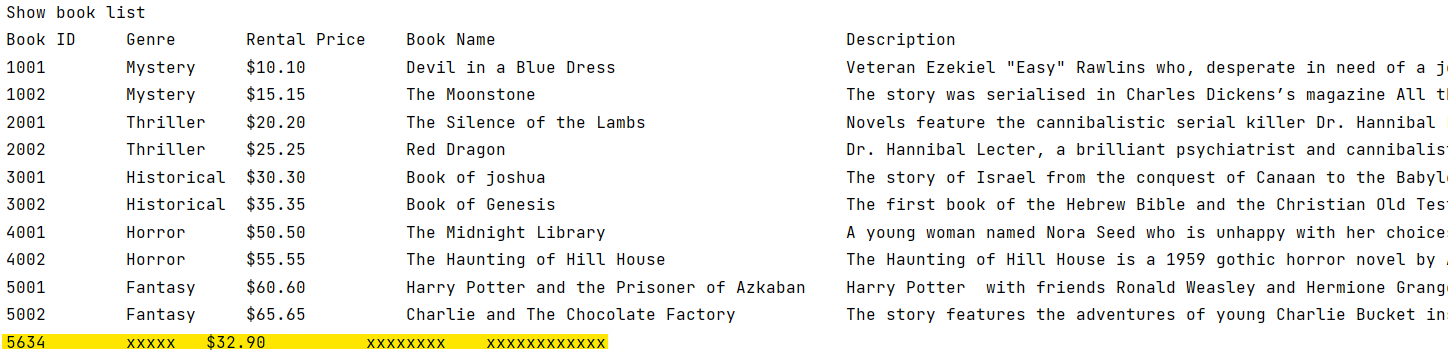
1. To move this application to enterprise level, we need to use a database and make this a web application with a web server.
2. To build a GUI for the application so that it is easier for visualisation and navigation, especially for the function to remove rental records.

The program lists all the rental records for the admin to choose and then enter the rental Id to remove.

The list will be very long for actual implementation. Admin would need to scroll for a long list to search for the rental ID to remove.

It should allow user to choose by clicking on checkboxes to select user account to remove.

1. In real life, book names can contain non ASCII characters such as Chinese book names, the program should be able to handle other non ASCII characters book names.
2. Since the output is in form of command line, display of output in an organised format is challenging. I use tab to print the different elements in columns. However if the name or description is too long or too short, the tab will not be aligned and the result will not be shown in a neat column. It will be good if this can display like a table in html.



# Appendix A: Java code



## Book.java

|  |
| --- |
| package sim.java.brm;  */\*\*  \* @author Jeslyn  \*  \*/* public class Book {  private int bookId;  private String bookName;  private String bookDescription;  private String bookGenre;  private double bookRentalPrice;   public Book(int bookId, String bookGenre, double bookRentalPrice, String bookName, String bookDescription) {  this.bookId = bookId;  this.bookGenre = bookGenre;  this.bookRentalPrice = bookRentalPrice;  this.bookName = bookName;  this.bookDescription = bookDescription;   }   public int getBookId() {  return bookId;  }   public void setBookId(int bookId) {  this.bookId = bookId;  }   public String getBookName() {  return bookName;  }   public void setBookName(String bookName) {  this.bookName = bookName;  }   public String getBookDescription() {  return bookDescription;  }   public void setBookDescription(String bookDescription) {  this.bookDescription = bookDescription;  }   public String getBookGenre() {  return bookGenre;  }   public void setBookGenre(String bookGenre) {  this.bookGenre = bookGenre;  }   public double getBookRentalPrice() {  return bookRentalPrice;  }   public void setBookRentalPrice(double bookRentalPrice) {  this.bookRentalPrice = bookRentalPrice;  } } |

## UserAccount.java

|  |
| --- |
| package sim.java.brm;  import java.util.ArrayList;  */\*\*  \* @author Jeslyn  \*  \*/* public class UserAccount {  private int userId;  private String userFirstName;  private String userLastName;  private String userCreateDate;  private ArrayList<String> userGenrePreferencesList;   public UserAccount(int userId, String userFirstName, String userLastName, String userCreateDate) {  this.userId = userId;  this.userFirstName = userFirstName;  this.userLastName = userLastName;  this.userCreateDate = userCreateDate;  this.userGenrePreferencesList = new ArrayList<>();  }   public void addGenre(String genre) {  this.userGenrePreferencesList.add(genre);  }   public void removeGenre(String genre) {  this.userGenrePreferencesList.remove(genre);  }   public int getUserId() {  return userId;  }   public void setUserId(int userId) {  this.userId = userId;  }   public String getUserFirstName() {  return userFirstName;  }   public void setUserFirstName(String userFirstName) {  this.userFirstName = userFirstName;  }   public String getUserLastName() {  return userLastName;  }   public void setUserLastName(String userLastName) {  this.userLastName = userLastName;  }   public String getUserCreateDate() {  return userCreateDate;  }   public void setUserCreateDate(String userCreateDate) {  this.userCreateDate = userCreateDate;  }   public ArrayList<String> getUserGenrePreferencesList() {  return userGenrePreferencesList;  }   public void setUserGenrePreferencesList(ArrayList<String> userGenrePreferencesList) {  this.userGenrePreferencesList = userGenrePreferencesList;  }  } |

## Rental.java

|  |
| --- |
| package sim.java.brm;  import java.util.ArrayList;  */\*\*  \* @author Jeslyn  \*  \*/* public class Rental {  private int rentalId;  private String rentalDate;  private UserAccount rentalUserAccount;  private ArrayList<Book> rentalBookList;   public Rental(int rentalId, String rentalDate, UserAccount rentalUserAccount) {  this.rentalId = rentalId;  this.rentalDate = rentalDate;  this.rentalUserAccount = rentalUserAccount;  this.rentalBookList = new ArrayList<>();  }   public void addBook(Book book) {  this.rentalBookList.add(book);  }   public void removeBook(Book book) {  this.rentalBookList.remove(book);  }   public double getTotalRentalPrice() {  double totalRentalPrice = 0;   for (Book book : rentalBookList) {  totalRentalPrice += book.getBookRentalPrice();  }   *// for(int i=0; i<rentalBookList.size(); i++) { // Book book = rentalBookList.get(i); // totalRentalPrice += book.getBookRentalPrice(); // }* return totalRentalPrice;  }   public Book getBook(int bookId) {  for(Book book : rentalBookList) {   if(bookId == book.getBookId()) {  return book;  }  }  return null;  }     public int getRentalId() {  return rentalId;  }   public void setRentalId(int rentalId) {  this.rentalId = rentalId;  }   public String getRentalDate() {  return rentalDate;  }   public void setRentalDate(String rentalDate) {  this.rentalDate = rentalDate;  }   public UserAccount getRentalUserAccount() {  return rentalUserAccount;  }   public void setRentalUserAccount(UserAccount rentalUserAccount) {  this.rentalUserAccount = rentalUserAccount;  }   public ArrayList<Book> getRentalBookList() {  return rentalBookList;  }   public void setRentalBookList(ArrayList<Book> rentalBookList) {  this.rentalBookList = rentalBookList;  } } |

## AdminAccount.java

|  |
| --- |
| package sim.java.brm;  */\*\*  \* @author Jeslyn  \*  \*/* public class AdminAccount {   private int adminId;  private String adminName;  private String adminUserName;  private String adminPassword;   public AdminAccount(int adminId, String adminName, String adminUserName, String adminPassword) {  this.adminId = adminId;  this.adminName = adminName;  this.adminUserName = adminUserName;  this.adminPassword = adminPassword;  }   public int getAdminId() {  return adminId;  }   public void setAdminId(int adminId) {  this.adminId = adminId;  }   public String getAdminName() {  return adminName;  }   public void setAdminName(String adminName) {  this.adminName = adminName;  }   public String getAdminUserName() {  return adminUserName;  }   public void setAdminUserName(String adminUserName) {  this.adminUserName = adminUserName;  }   public String getAdminPassword() {  return adminPassword;  }   public void setAdminPassword(String adminPassword) {  this.adminPassword = adminPassword;  }  } |

## ManagementSystem.java

|  |
| --- |
| package sim.java.brm;  import java.io.File; import java.io.FileNotFoundException; import java.text.DecimalFormat; import java.text.ParseException; import java.text.SimpleDateFormat; import java.util.ArrayList; import java.util.InputMismatchException; import java.util.List; import java.util.Scanner;  */\*\*  \* @author Jeslyn  \*  \*/* public class ManagementSystem {   private static List<AdminAccount> *adminAccountList* = new ArrayList<>();  private static List<Rental> *rentalList* = new ArrayList<>();  private static List<UserAccount> *userAccountList* = new ArrayList<>();  private static List<Book> *bookList* = new ArrayList<>();  private static AdminAccount *currentLoginAdminAccount* = null;  static Scanner *scanner*;  static Scanner *sc*;  public static void main(String[] args) {  *scanner* = new Scanner(System.*in*);  *sc* = new Scanner(System.*in*);  try {  *loadAdminAccountList*();  *loadBookList*();  *loadUserAccountList*();  *loadRentalList*();    *adminAccountSignIn*();   *displayMenu*();   while (*scanner*.hasNextLine()) {  String input = *scanner*.next();  try {   switch (input) {  case "1":  System.*out*.println("Add admin account");  *addAdminAccount*();  break;  case "2":  System.*out*.println("Delete admin account");   *deleteAdminAccount*();  break;  case "3":  System.*out*.println("Show admin account list");   *showAdminAccountList*();  break;  case "4":  System.*out*.println("Add book");  *addBook*();  break;  case "5":  System.*out*.println("Delete book");   *deleteBook*();  break;  case "6":  System.*out*.println("Show book list");  *showBookList*();  break;  case "7":  System.*out*.println("Most expensive book rental");  *getMostExpensiveBookRental*();  break;  case "8":  System.*out*.println("Cheapest book rental");  *getMostCheapestBookRental*();  break;  case "9":  System.*out*.println("Add user account");  *addUserAccount*();  break;  case "10":  System.*out*.println("Delete user account");  *deleteUserAccount*();  break;  case "11":  System.*out*.println("Show user account list");  *showUserAccountList*();  break;  case "12":  System.*out*.println("List all user rental records");  *listAllUserRentalRecords*();  break;  case "13":  System.*out*.println("Add user rental records");  *addUserRentalRecords*();  break;  case "14":  System.*out*.println("Remove user rental records");  *removeUserRentalRecords*();  break;    case "15":  System.*out*.println("Show top user that rent the most book");  *topUserThatRentTheMostBook*();  break;  case "16":  System.*out*.println("");  System.*out*.println("Show individual user matched books (Based on their genre preferences)");  *showIndividualUserMatchedBooks*();  break;   case "m":  case "M":  *displayMenu*();  break;  case "e":  case "E":  System.*out*.println("Exit");  System.*out*.println("Thank you and goodbye."); *// scanner.close(); // System.exit(0);* break;*//Exit use break so that program exit gracefully by rung finally to set scanner to close.* default:  System.*out*.println("Unrecognized option");  break;  }  if ("e".equals(input) || "E".equals(input) )  break;   } catch (InputMismatchException ime) {  System.*out*.println("Exception Encounter: integer expected. Please enter an integer");  } catch (NumberFormatException nfe) {  System.*out*.println("Exception Encounter: integer expected. Please enter an integer");  }   if (!"m".equals(input) && !"M".equals(input) ) {   System.*out*.print("Option Selection? ");  }  }  } catch (NumberFormatException nfe) {  System.*out*.println("Exception Encounter: integer expected. Please enter an integer");  } catch (Exception e) {  System.*out*.println("Exception: " + e);  } finally {  *scanner*.close();  *sc*.close();  System.*out*.println("\*\*\*\*\*\*\*Program End\*\*\*\*\*\*\*\*\*");  }  }   private static void adminAccountSignIn() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Bookkeep Rental Management \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Please sign in! (case sensitive)");  System.*out*.print("Admin User name: ");  String userName = *sc*.nextLine();  System.*out*.print("Admin Password: ");  String password = *sc*.nextLine();    for (AdminAccount adminAccount : *adminAccountList*) {  if (userName != null && userName.equals(adminAccount.getAdminUserName()) && password != null && password.equals(adminAccount.getAdminPassword())) {  *currentLoginAdminAccount* = adminAccount;  System.*out*.println("");  System.*out*.println("Welcome " + adminAccount.getAdminName() + "!");  }  }  System.*out*.println("");  if(*currentLoginAdminAccount* == null) {  System.*out*.println("Invaid Admin User Name or Password!");   *adminAccountSignIn*();  }   }   private static void displayMenu() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("\* MENU OPTION \*");  System.*out*.println("\* Please select an option: \*");  System.*out*.println("\* 1. Add admin account \*");  System.*out*.println("\* 2. Delete admin account \*");  System.*out*.println("\* 3. Show admin account list \*");   System.*out*.println("\* 4. Add book \*");  System.*out*.println("\* 5. Delete book \*");  System.*out*.println("\* 6. Show book list \*");  System.*out*.println("\* 7. Most expensive book rental \*");  System.*out*.println("\* 8. Cheapest book rental \*");   System.*out*.println("\* 9. Add user account \*");  System.*out*.println("\* 10. Delete user account \*");  System.*out*.println("\* 11. Show user account list \*");   System.*out*.println("\* 12. List all user rental records \*");  System.*out*.println("\* 13. Add user rental records \*");  System.*out*.println("\* 14. Remove user rental records \*");  System.*out*.println("\* 15. Show top user that rent the most books \*");  System.*out*.println("\* 16. Show individual user matched books (Based on their genre preferences) \*");  System.*out*.println("\* M. Display Menu \*");  System.*out*.println("\* E. Exit \*");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.print("Option Selection? ");   }   private static void loadAdminAccountList() {  System.*out*.println("[Method]: loadAdminAccountList");  Scanner sc = null;  try {  sc = new Scanner(new File("src/AdminAccountList.txt"));   sc.useDelimiter(System.*getProperty*("line.separator"));   while (sc.hasNext()) {  String line = sc.next();  System.*out*.println("Admin Account: " + line);  String[] lineArray = line.split("\\|");  if (line.startsWith("#")) {  *// Skip* } else {  AdminAccount adminAccount = new AdminAccount(Integer.*parseInt*(lineArray[0]), lineArray[1], lineArray[2], lineArray[3]);  *adminAccountList*.add(adminAccount);  }  }  System.*out*.println("");  } catch (FileNotFoundException e) {  System.*out*.print("Exception Encounter: src/AdminAccountList.txt file not found!");  return;  } finally {  sc.close();  }  }    private static void loadBookList() {  System.*out*.println("[Method]: loadBookList");  Scanner sc = null;  try {  sc = new Scanner(new File("src/BookList.txt"));   sc.useDelimiter(System.*getProperty*("line.separator"));   while (sc.hasNext()) {  String line = sc.next();  System.*out*.println("Book: " + line);  String[] lineArray = line.split("\\|");  if (line.startsWith("#")) {  *// Skip* } else {  Book book = new Book(Integer.*parseInt*(lineArray[0]), lineArray[1], Double.*parseDouble*(lineArray[2]), lineArray[3], lineArray[4]);  *bookList*.add(book);  }  }  System.*out*.println("");  } catch (FileNotFoundException e) {  System.*out*.print("Exception Encounter: src/BookList.txt file not found!");  return;  } finally {  sc.close();  }  }    private static void loadUserAccountList() {  System.*out*.println("[Method]: loadUserAccountList");  Scanner sc = null;  try {  sc = new Scanner(new File("src/UserAccountList.txt"));   sc.useDelimiter(System.*getProperty*("line.separator"));   while (sc.hasNext()) {  String line = sc.next();  System.*out*.println("Use Account: " + line);  String[] lineArray = line.split("\\|");  if (line.startsWith("#")) {  *// Skip* }   else if (line.startsWith("U")) {  UserAccount userAccount = new UserAccount(Integer.*parseInt*(lineArray[1]), lineArray[2], lineArray[3], lineArray[4]);  *userAccountList*.add(userAccount);  }   else if (line.startsWith("P")) {  int userId = Integer.*parseInt*(lineArray[1]);  UserAccount userAccount = *getUserAccount*(userId);  userAccount.addGenre(lineArray[2]);  }  }  System.*out*.println("");  } catch (FileNotFoundException e) {  System.*out*.print("Exception Encounter: src/UserAccountList.txt file not found!");  return;  } finally {  sc.close();  }  }    private static void loadRentalList() {  System.*out*.println("[Method]: loadRentalList");  Scanner sc = null;  try {  sc = new Scanner(new File("src/rentalBookList.txt"));   sc.useDelimiter(System.*getProperty*("line.separator"));   while (sc.hasNext()) {  String line = sc.next();  System.*out*.println("Use Account: " + line);  String[] lineArray = line.split("\\|");     if (line.startsWith("#")) {  *// Skip* }   else if (line.startsWith("R")) {  int rentalId = Integer.*parseInt*(lineArray[1]);  int userId = Integer.*parseInt*(lineArray[3]);  UserAccount userAccount = *getUserAccount*(userId);    if(userAccount == null)  System.*out*.print("Error Encounter in loading Rental list: UserId not found"+ userId);  else {  Rental rental = new Rental(rentalId, lineArray[2], userAccount);  *rentalList*.add(rental);  }  }   else if (line.startsWith("B")) {  int rentalId = Integer.*parseInt*(lineArray[1]);  Rental rental = *getRentalRecord*(rentalId);    if(rental == null)  System.*out*.print("Error Encounter in loading Rental list: rentalId not found"+ rentalId);  else {   int bookId = Integer.*parseInt*(lineArray[2]);  Book book = *getBook*(bookId);  if(book == null)  System.*out*.print("Error Encounter in loading Rental list: bookId not found"+ bookId);  else {  rental.addBook(book);  }  }  }  }  System.*out*.println("");  } catch (FileNotFoundException e) {  System.*out*.print("Exception Encounter: src/rentalBookList.txt file not found!");  return;  } finally {  sc.close();  }  }     private static void addAdminAccount() {  System.*out*.println("Please enter the below data item to create Admin Account");  System.*out*.print("Admin ID: " );    int adminId = Integer.*parseInt*(*sc*.nextLine());  if(*getAdminAccount*(adminId) != null) {  System.*out*.println("Error Encounter, adminId already exist!");  return;  }   System.*out*.print("Admin Name: " );  String adminName = *sc*.nextLine();  System.*out*.print("Admin User Name: " );  String adminUserName = *sc*.nextLine();  System.*out*.print("Admin Password: " );  String adminPassword = *sc*.nextLine();   AdminAccount adminAccount = new AdminAccount(adminId, adminName, adminUserName, adminPassword);  *adminAccountList*.add(adminAccount);  System.*out*.println("Record Added Successfully!");   }   private static void deleteAdminAccount() {  System.*out*.print("What is the Admin ID to be deleted? ");  int adminId = Integer.*parseInt*(*scanner*.next());  AdminAccount adminAccount = *getAdminAccount*(adminId);  if(adminAccount == null) {  System.*out*.println("Admin ID Not Found!");  return;  }  *adminAccountList*.remove(adminAccount);  System.*out*.println("Record Deleted Successfully!");    }   private static void showAdminAccountList() {  System.*out*.println("Sno\tAdmin Id\tAdmin Name\tAdmin User Name\t\tAdmin Password");  int i = 0;  for (AdminAccount adminAccount : *adminAccountList*) {  System.*out*.println(++i + "\t"+ adminAccount.getAdminId()+"\t\t" + adminAccount.getAdminName()+"\t\t" + adminAccount.getAdminUserName()+"\t\t\t" + adminAccount.getAdminPassword());  }  System.*out*.println("adminAccountList.size: "+ *adminAccountList*.size());  System.*out*.println("");  }   private static void addBook() {  System.*out*.println("Please enter the below data item to create Book");    System.*out*.print("Book ID: ");  int bookId = Integer.*parseInt*(*sc*.nextLine());  if(*getBook*(bookId) != null) {   System.*out*.println("Error Encounter, bookId already exist!");  return;  }  System.*out*.print("Book Name: ");  String bookName = *sc*.nextLine();  System.*out*.print("Book Description: ");  String bookDescription = *sc*.nextLine();  System.*out*.print("Book Genre: " );  String bookGenre = *sc*.nextLine();  System.*out*.print("bookRentalPrice: $");  double bookRentalPrice = Double.*parseDouble*(*sc*.next());     Book book = new Book(bookId, bookGenre, bookRentalPrice, bookName, bookDescription);  *bookList*.add(book);  System.*out*.println("Record Added Successfully!");  }   private static void deleteBook() {  System.*out*.print("What is the book ID to be deleted? ");  int bookId = Integer.*parseInt*(*scanner*.next());  Book book = *getBook*(bookId);  if(book == null) {  System.*out*.println("Book ID Not Found!");  return;  }   *bookList*.remove(book);  System.*out*.println("Record Deleted Successfully!");   }   private static void showBookList() {  System.*out*.println("Book ID\t\tGenre\t\tRental Price\tBook Name\t\t\t\t\t\t\t\t\tDescription");    for (Book item : *bookList*) {  *showBook*(item);  }  System.*out*.println("bookList.size: "+ *bookList*.size());  System.*out*.println("");  }   private static void showBook(Book book) {  DecimalFormat df = new DecimalFormat("00.00");  System.*out*.println(book.getBookId()+"\t\t" + book.getBookGenre() +"\t$"+ df.format(book.getBookRentalPrice()) +"\t\t\t"+ book.getBookName() +"\t"+ book.getBookDescription() );  }   private static void getMostExpensiveBookRental() {  Book mostExpensiveBookRental = null;   for (Book item : *bookList*) {  if (mostExpensiveBookRental == null)  mostExpensiveBookRental = item;  else if (mostExpensiveBookRental.getBookRentalPrice() < +item.getBookRentalPrice()) {  mostExpensiveBookRental = item;  }  }  System.*out*.println("Book ID\t\tGenre\t\tRental Price\tBook Name\t\t\t\t\t\t\t\t\tDescription");  *showBook*(mostExpensiveBookRental);  }   private static void getMostCheapestBookRental() {  Book mostCheapestBookRental = null;   for (Book item : *bookList*) {  if (mostCheapestBookRental == null)  mostCheapestBookRental = item;  else if (mostCheapestBookRental.getBookRentalPrice() > +item.getBookRentalPrice()) {  mostCheapestBookRental = item;  }  }  System.*out*.println("Book ID\t\tGenre\t\tRental Price\tBook Name\t\t\t\t\t\t\t\t\tDescription");  *showBook*(mostCheapestBookRental);  }   private static void addUserAccount() {   System.*out*.println("Please enter the below data item to Add User Account");  System.*out*.print("User ID: ");  int userId = Integer.*parseInt*(*sc*.nextLine());    UserAccount userAccount = *getUserAccount*(userId);   if(*getUserAccount*(userId) != null) {  System.*out*.println("Error Encounter, userId already exist!");  return;  }  System.*out*.print("User First Name: ");  String userFirstName = *sc*.nextLine();  System.*out*.print("User Last Name: ");  String userLastName = *sc*.nextLine();  System.*out*.print("User Create Date in dd/MM/yyyy format: " );  String userCreateDate = *sc*.nextLine();    if(userCreateDate.length() != 10) {  System.*out*.println("Error Encounter, invald date!");  return;  }    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");  try {  sdf.parse(userCreateDate);   }  catch (ParseException e) {  System.*out*.println(userCreateDate+" is Invalid Date format");  return;  }    userAccount = new UserAccount(userId, userFirstName, userLastName, userCreateDate);    *userAccountList*.add(userAccount);    while (*scanner*.hasNextLine()) {  System.*out*.print("Genre Preference (if no more genre preference enter 'N'): ");  String genre = *sc*.nextLine();  if("N".equalsIgnoreCase(genre) )  break;    userAccount.addGenre(genre);  }   System.*out*.println("Record Added Successfully!");    }   private static void deleteUserAccount() {    System.*out*.print("What is the User ID to be deleted? ");  int userId =Integer.*parseInt*(*scanner*.next());  UserAccount userAccount = *getUserAccount*(userId);    if(userAccount == null) {  System.*out*.println("User ID Not Found!");  return;  }    *userAccountList*.remove(userAccount);  System.*out*.println("Record Deleted Successfully!");   }    private static void showUserAccountList() {  System.*out*.println("");  System.*out*.println("User ID\t\tUser First Name\t\tUser Last Name\t\tUser Create Date");  for (UserAccount item : *userAccountList*) {  *showUserAccount*(item);  System.*out*.println("");  }  }   private static void showUserAccount(UserAccount userAccount) {  System.*out*.println(userAccount.getUserId() + "\t\t" + userAccount.getUserFirstName() + "\t\t\t" + userAccount.getUserLastName() + "\t\t\t" + userAccount.getUserCreateDate());  List<String> userGenrePreferencesList = userAccount.getUserGenrePreferencesList();  for (String item : userGenrePreferencesList) {  System.*out*.println("Genre Preferences: " + item);  }  }   private static void listAllUserRentalRecords() {  for (Rental rental : *rentalList*) {  *listUserRentalRecords*(rental);  System.*out*.println("");  }  }    private static void listUserRentalRecords(Rental rental) {  DecimalFormat df = new DecimalFormat("00.00");    UserAccount userAccount = rental.getRentalUserAccount();   List<Book> rentalBookList = rental.getRentalBookList();  System.*out*.println("Rental ID\tRental Date\tUser ID\t\tUser Name");  System.*out*.println(rental.getRentalId() + "\t\t" + rental.getRentalDate() + "\t" + userAccount.getUserId() + "\t\t" + userAccount.getUserFirstName());    System.*out*.println("\tBook ID\t\tRental Price\tGenre\t\tName");  for (Book book : rentalBookList) {  System.*out*.println("\t"+book.getBookId() + "\t\t$" +df.format(book.getBookRentalPrice()) + "\t\t\t" + book.getBookGenre() + "\t" + book.getBookName());  }  System.*out*.println("Total Rental Price: $"+ df.format(rental.getTotalRentalPrice()));  }    private static void topUserThatRentTheMostBook() {  Rental result = null;  for (Rental rental : *rentalList*) {  if (result == null)  result = rental;  else if (rental.getRentalBookList().size() > result.getRentalBookList().size())  result = rental;  }  *listUserRentalRecords*(result);  }   private static void showIndividualUserMatchedBooks() {  for(UserAccount userAccount : *userAccountList*) {    System.*out*.println("User: (" + userAccount.getUserId() + ") " + userAccount.getUserFirstName() + " " + userAccount.getUserLastName());  List<String> userGenrePreferencesList = userAccount.getUserGenrePreferencesList();  System.*out*.println("User Genre Preferences: ");  int i=0;  for (String item : userGenrePreferencesList) {  System.*out*.println(++i +") "+item);  }    System.*out*.println("Below are the books that matches the user genre preference:");  for (Book book : *bookList*) {  for (String genre : userGenrePreferencesList) {  if (genre.trim().equals(book.getBookGenre().trim())) {  System.*out*.println(book.getBookName() + " Genre(" + book.getBookGenre().trim()+")");  }  }  }  System.*out*.println("");  }  }    private static void addUserRentalRecords() { *// System.out.println("Rental Records List:"); // listAllUserRentalRecords(); // System.out.println("Book List:"); // showBookList();* System.*out*.print("\nPlease provide the rental ID: ");  int rentalId = Integer.*parseInt*(*scanner*.next());  Rental rental = *getRentalRecord*(rentalId);  if(rental == null) {  System.*out*.println("Error Encounter, rentalId Not Found!");  *addUserRentalRecords*();  }    System.*out*.print("Please provide the Book ID to add to the rental records: ");  int bookId = Integer.*parseInt*(*scanner*.next());  Book book = *getBook*(bookId);  if(book == null) {;  System.*out*.println("Error Encounter, BookId Not Found!");  *addUserRentalRecords*();  }  if(rental.getBook(bookId) != null) {  System.*out*.println("Error Encounter, BookId already exist in rental record!");  *addUserRentalRecords*();  }    rental.addBook(book);    System.*out*.println("Book added to rental record Successfully!"); *// System.out.println("Rental Records List:"); // listUserRentalRecords(rental);* }    private static void removeUserRentalRecords() { *// System.out.println("Rental Records List:"); // listAllUserRentalRecords();* System.*out*.print("\nPlease provide the rental ID: ");  int rentalId = Integer.*parseInt*(*scanner*.next());  Rental rental = *getRentalRecord*(rentalId);  if(rental == null) {  System.*out*.println("Error Encounter, RentalId Not Found!");  *removeUserRentalRecords*();  }    System.*out*.print("Please provide the Book ID to remove from the rental records: ");  int bookId = Integer.*parseInt*(*scanner*.next());  Book book = *getBook*(bookId);  if(book == null) {  System.*out*.println("Error Encounter, Book Id Not Found!");  *removeUserRentalRecords*();  }  rental.removeBook(book);  System.*out*.println("Book removed from rental record Successfully!");   *// System.out.println("Rental Records List:"); // listUserRentalRecords(rental);* }    private static Book getBook(int bookId) {  for (Book book : *bookList*) {  if(bookId == book.getBookId()) {  return book;  }  }  return null;  }    private static Rental getRentalRecord(int rentalId) {  for (Rental rental : *rentalList*) {  if(rentalId == rental.getRentalId()) {  return rental;  }  }  return null;  }    private static UserAccount getUserAccount(int userId) {  for(UserAccount userAccount : *userAccountList*) {  if(userId == userAccount.getUserId()) {  return userAccount;  }  }  return null;  }    private static AdminAccount getAdminAccount(int adminId) {  for(AdminAccount adminAccount : *adminAccountList*) {  if(adminId == adminAccount.getAdminId()) {  return adminAccount;  }  }  return null;  }  } |