

**Subject: Object Oriented Programming**

**Subject code: IT602**

**Date of submission:5-2-2022**

**Due date of submission:5-2-2022**

**Group Name: Elite**

**Group No: 3**

**Teaching Asst: Naiya Patel**

**Student Names:**

**Dev Adnani 202212012**

**Rudrik Prajapati 202212013**

**Sanket Acharya 202212100**

**Yash Khatri 202212086**

**Topic: Develop a console-based application for a bookstore.**

**Question Statement**

Part I: User Login/ Register

* Application should start by showing two options: (1) Sign In (2) Register

In the case of Sign In, ask the user to provide his “username” and “password”. If they

match, let her in, not otherwise.

* In the case of Register, ask the user to provide his “full name”, “username”, and

“password”. In case of password, echo asterisk symbol (\*) in place of the typed character.

E.g., a user types “1@4”, you should display “\*\*\*”, but you will store the actual

password to match when she logs in again, Hint: You can use java.io.Console.readPassword() method. Carefully read its description from the Java 8 docs online.

* In case of Sign In, provide an option of “Forgot Password”. Ask the user for her name, if

it matches then, let her set a new password and redirect her to login again with the new

password.

Part II: Display Catalog

* As the user successfully logs in, the application should display a list of available books,

their points (equivalent to book price), and their status of availability (e.g., In-store,

Out-of-Stock, 5 copies available).

* At the end of above list, it should also show two options: (1) Show Profile, (2) Purchase,

(2) Logout

* In the case of the “Show Profile”, it should show a list of books that she has already

purchased and the balance points that she can use to purchase new books. Of course, at

the end, there should be two options: (1) Back to the Catalog, (2) Logout.

Part III: Make Transaction

* On selecting the “Purchase” option, show the catalog to the user and ask her to type the

Book ID she wants to purchase.

* In case of sufficient balance, send her the message of successful transaction. Direct her to

her profile, show the book details on top of her purchased books lists, and update her

balance points.

**File Structure**

src

* book(folder/package)
  + book.java
  + bookHelper.java
  + Availability.java
* user(folder/package)
  + user.java
  + userHelper.java

main.java

**Book.java**

package book;

public class Book {

private final int book\_id;

private int copies\_left;

private final double points;

private final String name;

private final String author;

private Availability availability;

// Constructor For Book Class

public Book(int book\_id, int copies\_left, double price, String name, String author, Availability availability) {

this.book\_id = book\_id;

this.copies\_left = copies\_left;

this.points = price;

this.name = name;

this.author = author;

this.availability = availability;

}

// Get - Book ID

public int getBook\_id() {

return book\_id;

}

//Decreasing Book Copies

public void setCopies\_left() {

--copies\_left;

}

/// Getting Book Copies Left Length

public int getCopies\_left() {

return copies\_left;

}

// Get - Book Price

public double getPoints() {

return points;

}

// Get - Book Name

public String getName() {

return name;

}

// Get - Book Author

public String getAuthor() {

return author;

}

// Get - Book Availability

public Availability getAvailability() {

return availability;

}

//Set - Setting Book Availability

public void setAvailability(Availability availability) {

this.availability = availability;

}

}

**Description**

* The class has instance variables for book ID, the number of copies left, price, name, author, and availability. The class has getters and setters for each of these instance variables and also a constructor which initializes the variables.

**BookHelper.java**

package book;

import java.util.ArrayList;

public class BookHelper {

// All Books Data

ArrayList<Book> books = new ArrayList<Book>();

public BookHelper() {

books.add(new Book(1, 3, 110.11, "Doglapan", "Ashneer", Availability.IN\_STORE));

books.add(new Book(2, 2, 111, "How To Make Friends", "Dev",Availability.IN\_STORE));

books.add(new Book(3, 0, 99, "Naruto - Tale Of Ninja", "Ichiraku", Availability.OUT\_OF\_STOCK));

books.add(new Book(4, 0, 99, "Kakashi - Tale Of Ninja", "Naruto", Availability.OUT\_OF\_STOCK));

books.add(new Book(5, 0, 102.00, "Red - Tale Of Red Ranger", "Grum", Availability.OUT\_OF\_STOCK));

}

// Gets Specific Book

public Book getSpecificBook(int index) {

return books.get(index);

}

// Display All Books

public void showBooks() {

System.out.println("Book ID " + " | " + "Book Name" + " | " + "Book Points" + " | " +

" Book Author " + " | " + "Book Availability " + " | " + "Book Copies Left");

for (Book book : books) {

System.out.println(book.getBook\_id() + " | " + book.getName() + " | " + book.getPoints() +

" | " + book.getAuthor() + " | " + book.getAvailability() + " | " + book.getCopies\_left());

}

}

/// Getting Total Books Length

public int getBooksLength() {

return books.size();

}

// Decrease Book Copies After We Buy The Book

public void decreaseBookCopies(int book\_id) {

if (books.get(book\_id).getCopies\_left() > 1) {

books.get(book\_id).setCopies\_left();

} else {

books.get(book\_id).setAvailability(Availability.OUT\_OF\_STOCK);

books.get(book\_id).setCopies\_left();

}

}

}

**Description**

* The code defines a class named BookHelper which contains an ArrayList of Book objects named books.
* In the constructor of the class, some Book objects are created and added to the books ArrayList.
* The getSpecificBook method returns a book object from the books ArrayList based on the index passed to the function.
* The showBooks method displays all the books present in the books ArrayList.
* The getBooksLength method returns the number of books present in the books ArrayList.
* The decreaseBookCopies method decreases the number of copies of a book available in the store. If the number of copies is more than one, it decreases the number of copies, otherwise it changes the availability of the book to "OUT\_OF\_STOCK".

**Availability.java**

package book;

public enum Availability {

IN\_STORE,

OUT\_OF\_STOCK,

}

**Description**

* The code defines an enumerated type named "Availability".
* The type has two possible values: "IN\_STORE" and "OUT\_OF\_STOCK".
* The "IN\_STORE" value represents that a book is available in the store.
* The "OUT\_OF\_STOCK" value represents that a book is not available in the store..

**User.java**

package user;

import book.Book;

import java.util.ArrayList;

public class User {

private String username, password;

private double points = 600;

// Generates Random ID Between 10 to 100

private final int uid = (int) (Math.random() \* (100 - 10 + 1) + 10);

ArrayList<Book> bookBought = new ArrayList<Book>();

//Returns All Books Bought By Users

public ArrayList<Book> getBookBought() {

return bookBought;

}

// Buying Books Function

public boolean buyBook(Book book)

{

bookBought.add(book);

return true;

}

//Get - UserName

public String getUsername() {

return username;

}

// Set Username

public void setUsername(String username) {

this.username = username;

}

// Get - Password

public String getPassword() {

return password;

}

// Set - Password

public void setPassword(String password) {

this.password = password;

}

// Get - User Points

public double getPoints() {

return points;

}

// Set - User Points

public void setPoints(double points) {

this.points = points;

}

// Get User ID

public int getUid() {

return uid;

}

}

**Description**

* The code defines a class named "User". The class has several private variables including "username", "password", "points", and "uid".
* The "uid" is a randomly generated number between 10 and 100.
* The class also has an ArrayList named "bookBought" that keeps track of the books bought by the user.
* The class has several public methods including "getBookBought", "buyBook", "getUsername", "setUsername", "getPassword", "setPassword", "getPoints", "setPoints", and "getUid".
* The "getBookBought" method returns the ArrayList of books bought by the user. The "buyBook" method adds a book to the ArrayList of books bought by the user.
* The "getUsername" and "setUsername" methods are used to get and set the username, respectively.
* The "getPassword" and "setPassword" methods are used to get and set the password, respectively.
* The "getPoints" and "setPoints" methods are used to get and set the user's points, respectively.
* The "getUid" method returns the user's ID.

**UserHelper.java**

package user;

import book.Book;

import java.util.ArrayList;

import java.util.HashMap;

public class UserHelper {

//Forget Password - Just Checking If Name Is Same As Old Or Not ,

If Yes We Change The Password , Else We Return False

public boolean forgetPassword(String uName, String newPass, HashMap<String, User> users) {

boolean userNameCheck = users.containsKey(uName);

if (userNameCheck) {

users.get(uName).setPassword(newPass);

return true;

} else {

return false;

}

}

// Passing User Object And Using Its Data To Display User Profile

public void displayUserProfile(User user) {

System.out.println("--------------------------- USER PROFILE-------------------------------- ");

System.out.println("Username : " + user.getUsername());

System.out.println("Points : " + user.getPoints());

ArrayList<Book> bookBought = user.getBookBought();

System.out.println("Books Bought : ");

System.out.println("Book ID " + " | " + "Book Name" + " | " + " Book Author ");

for (int i = 0; i < user.getBookBought().size(); i++) {

System.out.println(bookBought.get(i).getBook\_id() + " | " + bookBought.get(i).getName() + " | " + bookBought.get(i).getAuthor());

}

System.out.println("----------------------END OF USER PROFILE-------------------------------- ");

}

}

**Description**

* The code defines a class named "UserHelper" that contains two methods: "forgetPassword" and "displayUserProfile".
* The "forgetPassword" method takes three parameters: "uName" which is a string representing the username, "newPass" which is a string representing the new password, and "users" which is a HashMap of users. The method first checks if the given username exists in the HashMap. If the username exists, the password is changed to the new password and the method returns true. If the username does not exist, the method returns false.
* The "displayUserProfile" method takes a "User" object as a parameter and displays the user's profile information such as the username, points, and books bought. The books bought information includes the book's ID, name, and author. The display is printed to the console.

**Main.java**

import book.Availability;

import book.Book;

import book.BookHelper;

import user.User;

import user.UserHelper;

import java.io.Console;

import java.util.HashMap;

import java.util.Scanner;

import static java.lang.System.exit;

public class Main {

public static void print(String msg) {

System.out.println(msg);

}

public static String consolePass() {

Console console = System.console();

String tempPass = "";

if (console != null) {

char tx[] = console.readPassword("Enter Password : ");

tempPass = String.valueOf(tx);

if (tempPass.length() != 0) {

for (char c : tx) {

System.out.print("\*");

}

}

print("");

}

return tempPass;

}

public static void main(String[] args) {

boolean isRegistered = false, isLoggedIn = false;

HashMap<String, User> users = new HashMap<>();

User loggedUser = null;

UserHelper userHelper = new UserHelper();

BookHelper bookHelper = new BookHelper();

Scanner scanner = new Scanner(System.in);

String userName = "", password = "";

String fmt = "%2$5s %3$10s%n";

int choice = -1;

//// User Login And Password

while (true) {

while (!isLoggedIn || !isRegistered) {

print("Enter 1 For Registering");

print("Enter 2 For Login");

print("Enter 3 For Forget Password");

print("Enter Your Choice : ");

choice = scanner.nextInt();

switch (choice) {

case 1 -> {

print("Please Enter username:");

scanner.nextLine();

userName = scanner.nextLine();

password = consolePass();

User createUser = new User();

//Checking If Username Already Exists - Not Permitting Multiple Users With Same Username

boolean allowUser = users.containsKey(userName);

if (!allowUser) {

createUser.setUsername(userName);

createUser.setPassword(password);

users.put(userName, createUser);

} else {

print("Username Already Exists");

}

isRegistered = true;

}

case 2 -> {

if (isRegistered) {

print("Please Enter username:");

scanner.nextLine();

String uName = scanner.nextLine();

boolean userNameCheck = users.containsKey(uName);

if (userNameCheck) {

String pass = consolePass();

String userPass = users.get(userName).getPassword();

if (userPass.equals(pass)) {

isLoggedIn = true;

loggedUser = users.get(userName);

} else {

print("Wrong userName or Password");

}

}

} else {

print("Please Register Before Trying To Login");

}

}

case 3 -> {

if (isRegistered) {

print("Please Enter username:");

scanner.nextLine();

String uName = scanner.nextLine();

String newPass = consolePass();

boolean passChangeStatus = userHelper.forgetPassword(uName, newPass, users);

if (passChangeStatus) {

print("Password Changed Successfully");

} else {

print("Oops Wrong Username");

}

} else {

print("You Must Register Before , If You Wanna Change Password");

}

}

default -> {

print("Wrong Choice...");

}

}

}

// Part 2 -> Display Books & Buy Books

while (choice != 3) {

bookHelper.showBooks();

print("\n \n");

print("Enter 1 For Show Profile");

print("Enter 2 For Purchase");

print("Enter 3 For Logout");

print("Enter 4 For Exit");

choice = scanner.nextInt();

switch (choice) {

case 1 -> {

// Show Profile

userHelper.displayUserProfile(loggedUser);

}

case 2 -> {

// Buy Books

print("Enter Book ID : ");

int ch = scanner.nextInt();

ch = ch - 1; // Decreasing One As We Have 0 Base Indexing

// Checking If Book ID aka Index Is Valid Or Not

if (ch >= 0 && ch < bookHelper.getBooksLength()) {

Book bk = bookHelper.getSpecificBook(ch);

double userPoints = loggedUser.getPoints();

double bookPoint = bk.getPoints();

if (bk.getAvailability() == Availability.IN\_STORE && userPoints >= bookPoint) {

boolean isCompleted = loggedUser.buyBook(bk);

loggedUser.setPoints(userPoints - bookPoint);

bookHelper.decreaseBookCopies(ch);

if (isCompleted) {

System.out.println("--------------------------- Transaction Status -------------------------------- ");

print("Book Bought");

System.out.println("--------------------------- Transaction Status -------------------------------- ");

userHelper.displayUserProfile(loggedUser);

}

} else {

System.out.println("--------------------------- Transaction Status -------------------------------- ");

print("The Book Is Not Available To Buy Or You Have Less Points Kindly Check");

System.out.println("--------------------------- Transaction Status -------------------------------- ");

}

} else {

print("Please Enter Valid ID");

}

}

case 3 -> {

isLoggedIn = false;

loggedUser = null;

break;

}

case 4 -> {

scanner.close();

exit(1);

}

}

}

}

}

}

**Description**

* This is a Java program for a book store where users can register, log in, purchase books, and show their profile. It uses several classes such as User, Book, UserHelper, BookHelper to perform these actions.
* When the program starts, it allows the user to choose between registering, logging in, and forgetting password. If the user logs in, they can see the list of books and buy books.
* They can also show their profile. If they want to log out, they can do so. If they want to exit, they can also do that.

**How To Run The Project ?**

* Here are the steps to run an IntelliJ project on another system
* Copy the project files: Transfer all the necessary files from the source system to the target system. This includes the source code, libraries, and configuration files.
* Install IntelliJ IDEA: Download and install the latest version of IntelliJ IDEA on the target system.
* Import the project: Launch IntelliJ IDEA and select "File" > "Open" to import the project files into the IDE.
* Run the project: Select the main class or run configuration and click the "Run" button to run the project.

**Screenshot :**





