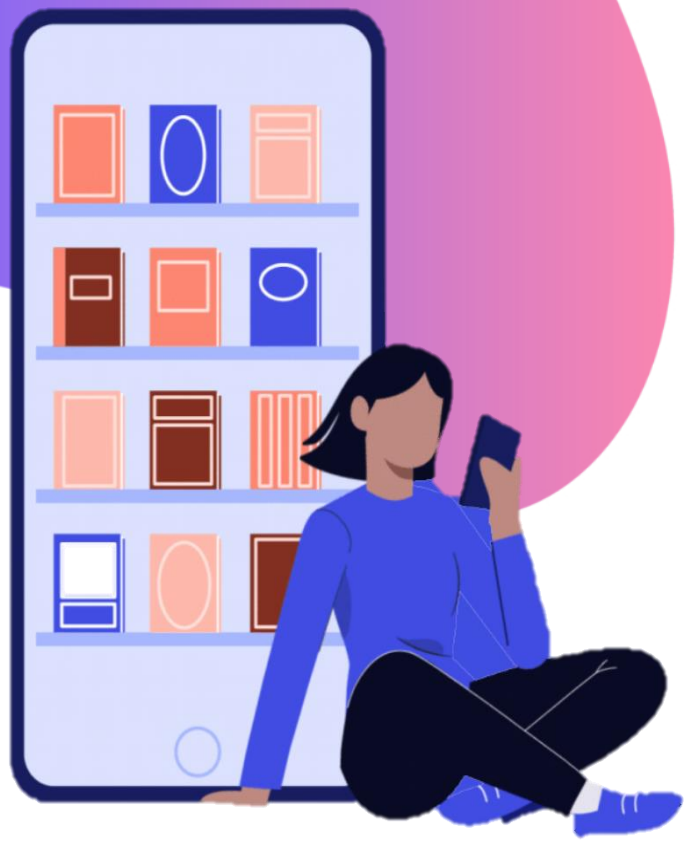
BOOK STORE

Online

@Short w fanila w Lap



Team leader

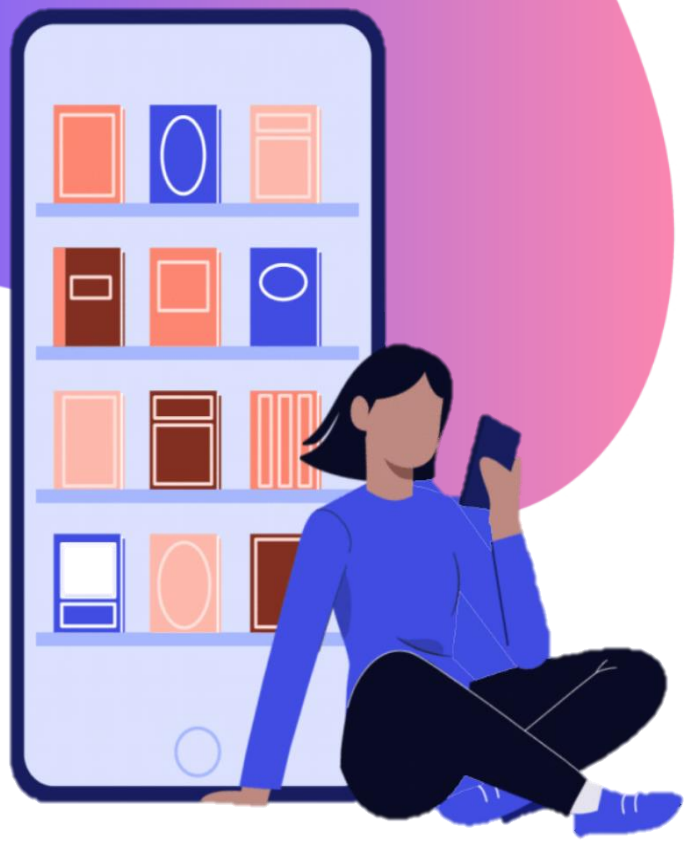
Barsoum Adly Daniel

202120014

@Short w fanila w Lap

Book Store

Team members



Barsoum Adly Daniel

202120014

Bishoy Labib Ezzat

202120131

Anton Rizk Gabriel

202120180

@Short w fanila w Lap

Book Store

```java
import main.view.menu.Menu;

public class Main {
    public static void main(String[] args) { Menu.startApplication(); }
}
```

class Main

Code

Book store

```java
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

abstract public class BookFileHandling {
    public final static String bookFilePath =
            "src/data/store/text/files/book_list.txt";

    public static void readFile() {
        try {
            File myFile = new File(bookFilePath);
            Scanner scanner = new Scanner(myFile);
            while (scanner.hasNextLine()) {
                ArrayList<String> bookData =
                        UserFileHandling.splitLine(scanner.nextLine());
                createBook(bookData);
            }
            scanner.close();
        } catch (IOException error) {
            error.printStackTrace();
        }
    }
```

Class BookFileHandling

Code

Book store

```java
public static void createBook(ArrayList<String> bookData) {
    Book book = new Book();
    book.setName(bookData.get(0));
    book.setAuthorName(bookData.get(1));
    book.setType(bookData.get(2));
    book.setPrice(bookData.get(3));
    Book.books.add(book);
}

public static void writeFile(ArrayList<String> data) {
    try {
        FileWriter myFile = new FileWriter(bookFilePath, append: true);
        myFile.write( str: '\n' + data.get(0) + ',' + data.get(1)
                + ',' + data.get(2) + ','+ data.get(3));
        myFile.close();
    } catch (IOException error) {
        error.printStackTrace();
    }
}
```

Code

Book store

Class BookFileHandling

```
    public static void rewriteFile(ArrayList<Book>books) {
        try {
            FileWriter myFile = new FileWriter(bookFilePath);
            for (Book book : books) {
                myFile.write( str: book.getName() + ',' + book.getAuthorName()
                        + ',' + book.getType() + ',' + book.getPrice() + '\n');
            }
            myFile.close();
        } catch (IOException error) {
            error.printStackTrace();
        }
    }
}
```

Class BookFileHandling

Code

Book store

```java
package data.store;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

abstract public class BorrowFileHandling {
    public final static String borrowFilePath
            = "src/data/store/text/files/borrowed_books.txt";
    public static final ArrayList<String> borrowKeys = new ArrayList<>();
    public static final HashMap<String, ArrayList<String>> borrowBooks
            = new HashMap<>();
```

Class BorrowFileHandling

Code

Book store

```java
public static void readFile() {
    try {
        File myFile = new File(borrowFilePath);
        Scanner scanner = new Scanner(myFile);
        while (scanner.hasNextLine()) {
            ArrayList<String> data =
                    UserFileHandling.splitLine(scanner.nextLine());
            addBookToBorrow(data);
        }
        scanner.close();
    } catch (IOException exception) {
        exception.printStackTrace();
    }
}

private static void addBookToBorrow(ArrayList<String> data) {
    ArrayList<String> bookList = new ArrayList<>();
    bookList.add(data.get(1));
    bookList.add(data.get(2));
    borrowBooks.put(data.get(0), bookList);
    borrowKeys.add(data.get(0));
}
```

Class BorrowFileHandling

Code

Book store

```java
public static String getKeyByCustomer(String customer) {
    for (String key : borrowKeys) {
        if (key.contains(customer)) {
            return key;
        }
    }
    return null;
}

public static void displayValueOfKey(String key) {
    ArrayList<String> books = borrowBooks.get(key);
    for (int i = 0; i < books.size(); i += 2) {
        System.out.println("\nBook name: " + books.get(i) +
                "\nPrice: " + books.get(i + 1) + "$"
                + "\n**---------------------------------**");
    }
}

public static ArrayList<String> getValueByKey(String key) { return borrowBooks.get(key); }
```

Code

Book store

Class BorrowFileHandling

```java
package data.store;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

abstract public class CartFileHandling {
    public final static String cartFilePath
            = "src/data/store/text/files/cart_books.txt";
    public static final HashMap<String, ArrayList<String>> cartBooks
            = new HashMap<>();
    public static final ArrayList<String> cartKeys = new ArrayList<>();
```

Code

Book store

Class CartFileHandling

```java
public static void readFile() {
    try {
        File myFile = new File(cartFilePath);
        Scanner scanner = new Scanner(myFile);
        while (scanner.hasNextLine()) {
            ArrayList<String> data =
                    UserFileHandling.splitLine(scanner.nextLine());
            addBookToCart(data);
        }
        scanner.close();
    } catch (IOException exception) {
        exception.printStackTrace();
    }
}

private static void addBookToCart(ArrayList<String> data) {
    ArrayList<String> books = new ArrayList<>();
    for (int i = 1; i < data.size(); i++) {
        books.add(data.get(i));
    }
    cartBooks.put(data.get(0), books);
    cartKeys.add(data.get(0));
}
```

Class CartFileHandling

Code

Book store

```
    public static String getKeyByCustomer(String customerName) {
        for (String key : cartKeys) {
            if (key.contains(customerName)) {
                return key;
            }
        }
        return null;
    }

    public static void displayValueOfKey(String key) {
        ArrayList<String> books = cartBooks.get(key);
        for (int i = 0; i < books.size(); i += 2) {
            System.out.println("\nBook name: " + books.get(i) +
                    "\nPrice: " + books.get(i + 1) + "$"
                    + "\n**--------------------------------**");
        }
    }

    public static ArrayList<String> getValueByKey(String key) { return cartBooks.get(key); }
}
```

Code

Book store

Class CartFileHandling

```java
package data.store;

import main.classes.Administrator;
import main.classes.Customer;
import main.classes.User;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

abstract public class UserFileHandling {
    public final static String customerFilePath =
            "src/data/store/text/files/customers_data.txt";
    public final static String administratorFilePath =
            "src/data/store/text/files/administrators_data.txt";
    public final static String newAdministratorFilePath =
            "src/data/store/text/files/inactivate_administrator_account.txt";
```

Class UserFileHandling

Code

Book store

```java
public static void readFile() {
    ArrayList<ArrayList<String>> customerData = getDataFromFile(
            customerFilePath);
    addCustomer(customerData);
    ArrayList<ArrayList<String>> administratorData = getDataFromFile(
            administratorFilePath);
    addAdministrator(administratorData, administratorFilePath);

    ArrayList<ArrayList<String>> newAdministratorData
            = getDataFromFile(newAdministratorFilePath);
    addAdministrator(newAdministratorData, newAdministratorFilePath);
}
```

Code

Book store

Class UserFileHandling

```java
private static ArrayList<ArrayList<String>>
getDataFromFile(String filePath) {
    ArrayList<ArrayList<String>> userData = new ArrayList<>();
    try {
        File myFile = new File(filePath);
        Scanner scanner = new Scanner(myFile);
        while (scanner.hasNextLine()) {
            ArrayList<String> data = splitLine(scanner.nextLine());
            userData.add(data);
        }
        scanner.close();
    } catch (IOException exception) {
        exception.printStackTrace();
    }
    return userData;
}
```

Code

Book store

Class UserFileHandling

```java
public static ArrayList<String> splitLine(String line) {
    ArrayList<String> userData = new ArrayList<>();
    StringBuilder data = new StringBuilder();
    for (int i = 0; i < line.length(); i++) {
        if (line.charAt(i) != ',') {
            data.append(line.charAt(i));
        } else {
            userData.add(data.toString());
            data = new StringBuilder();
        }
    }
    userData.add(data.toString());
    return userData;
}

public static void addCustomer(ArrayList<ArrayList
        <String>> customerData) {
    for (ArrayList<String> customerDatum : customerData) {
        Customer customer = new Customer();
        createUser(customer, customerDatum);
        Customer.customerData.add(customer);
    }
}
```

Code

Book store

Class UserFileHandling

```
public static void addAdministrator(ArrayList<ArrayList
        <String>> administratorData, String filePath) {
    for (ArrayList<String> administratorDatum : administratorData) {
        Administrator administrator = new Administrator();
        createUser(administrator, administratorDatum);
        if (filePath.equals(administratorFilePath)) {
            Administrator.administratorData.add(administrator);
        } else {
            Administrator.inactiveAdministratorData.add(administrator);
        }
    }
}

public static void createUser(User user, ArrayList<String> userData) {
    user.setFirstName(userData.get(0));
    user.setLastName(userData.get(1));
    user.setEmailAddress(userData.get(2));
    user.setPassword(userData.get(3));
    user.setPhone(userData.get(4));
}
```

Class UserFileHandling

Code

Book store

```java
public static void writeFile(String filePath, ArrayList<String> data) {
    try {
        FileWriter myFile;
        if (filePath.equals(administratorFilePath) ||
                filePath.equals(customerFilePath)) {
            myFile = new FileWriter(filePath, append: true);
            myFile.write( str: '\n' + data.get(0) + ',' + data.get(1) + ',' +
                    data.get(2) + ',' + data.get(3) + ',' + data.get(4));
        } else {
            myFile = new FileWriter(filePath, append: true);
            myFile.write( str: data.get(0) + ',' + data.get(1) + ',' +
                    data.get(2) + ',' + data.get(3) + ',' + data.get(4) + '\n');
        }
        myFile.close();
    } catch (IOException error) {
        System.out.println("An error occurred");
        error.printStackTrace();
    }
}
```

Code

Book store

Class UserFileHandling

```
    public static void rewriteInactiveAdministratorFile() {
        try {
            FileWriter myFile = new FileWriter(newAdministratorFilePath);
            ArrayList<User> newAdministrator
                    = Administrator.inactiveAdministratorData;
            for (User user : newAdministrator) {
                myFile.write( str: user.getFirstName() + ','
                        + user.getLastName() + ','
                        + user.getEmailAddress()
                        + ',' + user.getPassword() + ','
                        + user.getPassword() + '\n');
            }
            myFile.close();
        } catch (IOException error) {
            System.out.println("An error occurred");
            error.printStackTrace();
        }
    }
}
```

Code

Book store

Class UserFileHandling

```java
package main.classes;

import java.util.ArrayList;

public class Administrator extends User {
    public static ArrayList<User>
            administratorData = new ArrayList<>();
    public static ArrayList<User>
            inactiveAdministratorData = new ArrayList<>();

    public Administrator() { super(); }
}
```

Class Administrator

Code

Book store

Code

Book store

```java
package main.classes;

import java.util.ArrayList;

public class Customer extends User {
    public static ArrayList<User> customerData = new ArrayList<>();

    public Customer() { super(); }
}
```

Class Customer

```java
package main.classes;

import main.view.common.ConsoleReader;

import java.util.ArrayList;

public class User {
    private String firstName;
    private String lastName;
    private String emailAddress;
    private String password;
    private String phone;

    public User() {
        this.firstName = "Unknown name";
        this.lastName = "Unknown name";
    }

    public String getFirstName() { return firstName; }

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

    public String getLastName() { return lastName; }

    public void setLastName(String lastName) { this.lastName = lastName; }
```

Code

Book store

Class User

```java
public String getEmailAddress() { return emailAddress; }

public void setEmailAddress(String emailAddress) { this.emailAddress = emailAddress; }

public String getPassword() { return password; }

public void setPassword(String password) { this.password = password; }

public String getPhone() { return phone; }

public void setPhone(String phone) { this.phone = phone; }
```

Code

Book store

Class User

```java
public static String emailValidation(
        String emailAddress, ArrayList<User> userData) {
    int i = 0;
    String name = "No name";
    while (i < userData.size() && (!userData.get(i).getEmailAddress()
            .equals(emailAddress))) {
        i++;
    }
    if (i < userData.size()) {
        name = userData.get(i).firstName;
        passwordValidation(userData.get(i), ConsoleReader.readPassword());
    } else if (userData.equals(Administrator.administratorData)) {
        newAdministratorEmailValidation(emailAddress, userData);
    } else {
        System.out.println("\tNot found");
        emailValidation(ConsoleReader.readEmailAddress(), userData);
    }
    return name;
}
private static void passwordValidation(User user, String password) {
    if (!user.getPassword().equals(password)) {
        System.out.println("\tInvalid password");
        passwordValidation(user, ConsoleReader.readPassword());
    }
}
```

Code

Book store

Class User

```
private static void newAdministratorEmailValidation(
        String emailAddress, ArrayList<User> userData) {
    if (Administrator.inactiveAdministratorData.size() == 0) {
        System.out.println("\tNot found");
        emailValidation(ConsoleReader.readEmailAddress(), userData);
    } else {
        checkNewAdministratorEmail(emailAddress, userData);
    }
}
```

Code

Book store

Class User

```
    private static void checkNewAdministratorEmail(
            String emailAddress, ArrayList<User> userData) {
        ArrayList<User> newAdministrator
                = Administrator.inactiveAdministratorData;
        for (int j = 0; j < newAdministrator.size(); j++) {
            if (newAdministrator.get(j).
                    getEmailAddress().equals(emailAddress)) {
                passwordValidation(newAdministrator.get(j),
                        ConsoleReader.readPassword());
                System.out.println("\tYour email is still not activated");
                ConsoleReader.makeSpace();
                break;
            } else if (j == newAdministrator.size() - 1) {
                System.out.println("\tNot found");
                emailValidation(ConsoleReader.readEmailAddress(), userData);
            }
        }
    }
}
```

Class User

Code

Book store

```java
package main.classes;

import java.util.ArrayList;

public class Book {
    private String name;
    private String authorName;
    private String price;
    private String type;

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


    public String getPrice() { return price; }

    public void setPrice(String price) { this.price = price; }

    public String getName() { return name; }

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

    public String getAuthorName() { return authorName; }

    public void setAuthorName(String authorName) { this.authorName = authorName; }
```

Class Book

Code

Book store

```
    public String getType() { return type; }

    public void setType(String type) { this.type = type; }

    @Override
    public String toString() {
        return "\nBook name: " + name +
                "\nAuthor: " + authorName +
                "\nType: " + type + "\nPrice: " + price +
                "$\n**---------------------------------------**";
    }
}
```

Code

Book store

Class Book

```java
package main.view.administrator.options;

import data.store.BookFileHandling;
import main.classes.Book;
import main.view.common.CommonUtilities;
import main.view.common.ConsoleReader;
import main.view.customer.options.SearchUtilities;

import java.util.ArrayList;
import java.util.HashMap;

abstract public class AddUtilities {
    private enum Options {
        ADD_BOOK, RETURN_BACK, WRONG
    }

    private static final HashMap<String, Options>
            addOptions = new HashMap<>();

    private static void setMenuOptions() {
        addOptions.put("1", Options.ADD_BOOK);
        addOptions.put("2", Options.RETURN_BACK);
    }
```

Code

Book store

Class AddUtilities

```java
private static Options mapper(String option) {
    setMenuOptions();
    if (addOptions.get(option) == null) {
        return Options.WRONG;
    }
    return addOptions.get(option);
}

public static void displayOptionsMenu() {
    ConsoleReader.makeSpace();
    System.out.println("\t\t\t ***  Welcome " +
            AdministratorOptionList.administratorName + "  ***");
    System.out.println("\t1- Add a book");
    System.out.println("\t2- Return back");
    executeOption(ConsoleReader.getOption());
}
```

Code

Book store

Class AddUtilities

```
private static void executeOption(String option) {
    switch (mapper(option)) {
        case ADD_BOOK -> addBook();
        case RETURN_BACK -> CommonUtilities.
                returnBackToAdministratorMenu();
        default -> {
            System.out.println("\tInvalid option");
            ConsoleReader.makeSpace();
            displayOptionsMenu();
        }
    }
}

private static void addBook() {
    ArrayList<String> bookDetails = new ArrayList<>();
    bookDetails.add(ConsoleReader.readBookName());
    bookDetails.add(ConsoleReader.readAuthorName());
    bookDetails.add(ConsoleReader.readBookType());
    bookDetails.add(ConsoleReader.readBookPrice());
    BookFileHandling.createBook(bookDetails);
    BookFileHandling.writeFile(bookDetails);
    performAdd(Book.books.get(Book.books.size() - 1));
}
```

Code

Book store

Class AddUtilities

```
    private static void performAdd(Book book) {
        SearchUtilities.searchForBookName(book.getName());
        System.out.println("\tBook is added successfully");
        displayOptionsMenu();
    }
}
```

Class AddUtilities

Code

Book store

```java
package main.view.administrator.options;

import main.view.common.ConsoleReader;
import main.view.common.ShowUtilities;
import main.view.menu.Menu;

import java.util.HashMap;

abstract public class AdministratorOptionList {
    public static String administratorName;

    private enum Options {
        SHOW_BOOK_LIST, UPDATE_BOOK_DETAILS,
        ADD_BOOK, DELETE_BOOK, GIVE_ACCESS, LOGOUT, WRONG
    }

    private static final HashMap<String, Options>
            administratorOptions = new HashMap<>();
```

Class
AdministratorOptionList

Code

Book store

```
private static void setMenuOptions() {
    administratorOptions.put("1", Options.ADD_BOOK);
    administratorOptions.put("2", Options.SHOW_BOOK_LIST);
    administratorOptions.put("3", Options.UPDATE_BOOK_DETAILS);
    administratorOptions.put("4", Options.DELETE_BOOK);
    administratorOptions.put("5", Options.GIVE_ACCESS);
    administratorOptions.put("6", Options.LOGOUT);
}

private static Options mapper(String option) {
    setMenuOptions();
    if (administratorOptions.get(option) == null) {
        return Options.WRONG;
    }
    return administratorOptions.get(option);
}
```

Class
AdministratorOptionList

Code

Book store

```java
public static void displayOptionsMenu(
        String administratorName) {
    AdministratorOptionList.administratorName = administratorName;
    System.out.println("\t\t\t ***  Welcome " +
            administratorName + "  ***");
    System.out.println("\t1- Add a book");
    System.out.println("\t2- Show book list");
    System.out.println("\t3- Update book details");
    System.out.println("\t4- Delete a book");
    System.out.println("\t5- Give an access");
    System.out.println("\t6- Log out");
    executeOption(ConsoleReader.getOption());
}
```

Class
AdministratorOptionList

Code

Book store

```
    private static void executeOption(String option) {
        switch (mapper(option)) {
            case SHOW_BOOK_LIST -> ShowUtilities.displayOptionsMenu();
            case UPDATE_BOOK_DETAILS -> UpdateUtilities.
                    displayOptions();
            case ADD_BOOK -> AddUtilities.displayOptionsMenu();
            case DELETE_BOOK -> DeleteUtilities.displayOptionsMenu();
            case GIVE_ACCESS -> GiveAccessUtilities.displayOptionsMenu();
            case LOGOUT -> {
                ConsoleReader.makeSpace();
                Menu.displayMainView();
            }
            default -> {
                System.out.println("\tInvalid Option");
                ConsoleReader.makeSpace();
                displayOptionsMenu(administratorName);
            }
        }
    }
}
```

Class
AdministratorOptionList

Code

Book store

```java
package main.view.administrator.options;

import data.store.BookFileHandling;
import main.classes.Book;
import main.view.common.CommonUtilities;
import main.view.common.ConsoleReader;
import main.view.customer.options.SearchUtilities;

import java.util.HashMap;

abstract public class DeleteUtilities {
    private enum Options {
        DELETE_BOOK, RETURN_BACK, WRONG
    }

    private static final HashMap<String, Options>
            deleteOptions = new HashMap<>();

    private static void setMenuOptions() {
        deleteOptions.put("1", Options.DELETE_BOOK);
        deleteOptions.put("2", Options.RETURN_BACK);
    }
```

Code

Book store

Class DeleteUtilities

```java
private static Options mapper(String option) {
    setMenuOptions();
    if (deleteOptions.get(option) == null) {
        return Options.WRONG;
    }
    return deleteOptions.get(option);
}

public static void displayOptionsMenu() {
    ConsoleReader.makeSpace();
    System.out.println("\t\t\t ***  Welcome " +
            AdministratorOptionList.administratorName + "  ***");
    System.out.println("\t1- Delete a book");
    System.out.println("\t2- Return back");
    executeOption(ConsoleReader.getOption());
}
```

Code

Book store

Class DeleteUtilities

```
private static void executeOption(String option) {
    switch (mapper(option)) {
        case DELETE_BOOK -> deleteBook();
        case RETURN_BACK -> CommonUtilities.
                returnBackToAdministratorMenu();
        default -> {
            System.out.println("\tInvalid option");
            ConsoleReader.makeSpace();
            displayOptionsMenu();
        }
    }
}
```

Code

Book store

Class DeleteUtilities

```
    private static void deleteBook() {
        String bookName = ConsoleReader.readBookName();
        if (CommonUtilities.isBookExist(bookName)) {
            for (int i = 0; i < Book.books.size(); i++) {
                if (Book.books.get(i).getName().equals(bookName)) {
                    SearchUtilities.searchForBookName(Book.books.get(i).getName());
                    Book.books.remove(i);
                    performDelete();
                    break;
                }
            }
        } else {
            System.out.println("\tInvalid book name");
            deleteBook();
        }
    }

    private static void performDelete() {
        BookFileHandling.rewriteFile(Book.books);
        System.out.println("\tBook is deleted successfully");
        displayOptionsMenu();
    }
}
```

Code

Book store

Class DeleteUtilities

```java
package main.view.administrator.options;

import data.store.UserFileHandling;
import main.classes.Administrator;
import main.classes.User;
import main.view.common.CommonUtilities;
import main.view.common.ConsoleReader;

import java.util.ArrayList;
import java.util.HashMap;

abstract public class GiveAccessUtilities {
    private enum Options {
        GIVE_ACCESS, RETURN_BACK, WRONG
    }

    private static final HashMap<String, Options>
            giveAccessOptions = new HashMap<>();

    private static void setMenuOptions() {
        giveAccessOptions.put("1", Options.GIVE_ACCESS);
        giveAccessOptions.put("2", Options.RETURN_BACK);
    }
```

Code

Book store

Class GiveAccessUtilities

```
private static Options mapper(String option) {
    setMenuOptions();
    if (giveAccessOptions.get(option) == null) {
        return Options.WRONG;
    }
    return giveAccessOptions.get(option);
}

public static void displayOptionsMenu() {
    ConsoleReader.makeSpace();
    System.out.println("\t\t\t ***  Welcome " +
            AdministratorOptionList.administratorName + "  ***");
    System.out.println("\t1- Give an access");
    System.out.println("\t2- Return back");
    executeOption(ConsoleReader.getOption());
}
```

Class GiveAccessUtilities

Code

Book store

```
private static void executeOption(String option) {
    switch (mapper(option)) {
        case GIVE_ACCESS -> giveAccess();
        case RETURN_BACK -> CommonUtilities.
                returnBackToAdministratorMenu();
        default -> {
            System.out.println("\tInvalid option");
            ConsoleReader.makeSpace();
            displayOptionsMenu();
        }
    }
}
```

Class GiveAccessUtilities

Code

Book store

```
private static void giveAccess() {
    if (Administrator.inactiveAdministratorData.size() != 0) {
        int size = Administrator.inactiveAdministratorData.size();
        for (int i = 0; i < size; i++) {
            if (size > Administrator.inactiveAdministratorData.size()) {
                i--;
                size = Administrator.inactiveAdministratorData.size();
            }
            ArrayList<String> newAdministrators = getData(i);
            ConsoleReader.makeSpace();
            System.out.println(newAdministrators.get(2));
            performGivingAccess(newAdministrators, i);
        }
    } else {
        System.out.println("\tList is empty");
    }
    displayOptionsMenu();
}
```

Class GiveAccessUtilities

Code

Book store

```java
private static ArrayList<String> getData(int i) {
    ArrayList<String> newAdministrators = new ArrayList<>();
    ArrayList<User> administratorList
            = Administrator.inactiveAdministratorData;
    newAdministrators.add(administratorList.get(i).getFirstName());
    newAdministrators.add(administratorList.get(i).getLastName());
    newAdministrators.add(administratorList.get(i).getEmailAddress());
    newAdministrators.add(administratorList.get(i).getPassword());
    newAdministrators.add(administratorList.get(i).getPhone());
    return newAdministrators;
}
private static void performGivingAccess(
        ArrayList<String> newAdministrators, int index) {
    System.out.println("\t1- Give access");
    System.out.println("\t2- Give no access");
    switch (ConsoleReader.getOption()) {
        case "1" -> getAccess(newAdministrators, index);
        case "2" -> {
        }
        default -> {
            System.out.println("\tInvalid option");
            performGivingAccess(newAdministrators, index);
        }
    }
}
```

Class GiveAccessUtilities

Code

Book store

```
    private static void getAccess(
            ArrayList<String> newAdministrators, int index) {
        UserFileHandling.writeFile(UserFileHandling.
                administratorFilePath, newAdministrators);
        Administrator.administratorData.add(
                Administrator.inactiveAdministratorData.get(index));
        Administrator.inactiveAdministratorData.remove(index);
        UserFileHandling.rewriteInactiveAdministratorFile();
    }
}
```

Code

Book store

Class GiveAccessUtilities

```java
package main.view.administrator.options;

import main.classes.Book;
import data.store.BookFileHandling;
import main.view.common.CommonUtilities;
import main.view.common.ConsoleReader;
import main.view.customer.options.SearchUtilities;

import java.util.HashMap;

abstract public class UpdateUtilities {
    private enum Options {
        BOOK_NAME, BOOK_TYPE, AUTHOR_NAME,
        PRICE, RETURN_BACK, WRONG
    }

    private static final HashMap<String, Options>
            updateOptions = new HashMap<>();
```

Class UpdateUtilities

Code

Book store

```java
private static void setOptions() {
    updateOptions.put("1", Options.BOOK_NAME);
    updateOptions.put("2", Options.AUTHOR_NAME);
    updateOptions.put("3", Options.BOOK_TYPE);
    updateOptions.put("4", Options.PRICE);
    updateOptions.put("5", Options.RETURN_BACK);
}

private static Options mapper(String option) {
    setOptions();
    if (updateOptions.get(option) == null) {
        return Options.WRONG;
    }
    return updateOptions.get(option);
}
```

Class UpdateUtilities

Code

Book store

```
public static void displayOptions() {
    ConsoleReader.makeSpace();
    System.out.println("\t\t\t ***  Welcome " +
            AdministratorOptionList.administratorName + "  ***");
    System.out.println("\t1- Update book name");
    System.out.println("\t2- Update author name");
    System.out.println("\t3- Update book type");
    System.out.println("\t4- Update price");
    System.out.println("\t5- Return back");
    executeOptions(ConsoleReader.getOption());
}
private static void executeOptions(String option) {
    switch (mapper(option)) {
        case BOOK_NAME -> updateBookName();
        case AUTHOR_NAME -> updateAuthorName();
        case BOOK_TYPE -> updateBookType();
        case PRICE -> updatePrice();
        case RETURN_BACK -> CommonUtilities.
                returnBackToAdministratorMenu();
        default -> {
            System.out.println("\tInvalid Option");
            displayOptions();
        }
    }
}
```

Class UpdateUtilities

Code

Book store

```
private static void updateBookName() {
    Book book = getBook();
    String newBookName = ConsoleReader.readNewBookName();
    book.setName(newBookName);
    performUpdate( change: "Book name", book);
}

private static void updateBookType() {
    Book book = getBook();
    String newBookType = ConsoleReader.readNewBookType();
    book.setType(newBookType);
    performUpdate( change: "Book type", book);
}

private static void updateAuthorName() {
    Book book = getBook();
    String newAuthorName = ConsoleReader.readNewAuthorName();
    book.setAuthorName(newAuthorName);
    performUpdate( change: "Book author name", book);
}
```

Code

Book store

Class UpdateUtilities

```java
private static void updateBookName() {
    Book book = getBook();
    String newBookName = ConsoleReader.readNewBookName();
    book.setName(newBookName);
    performUpdate( change: "Book name", book);
}

private static void updateBookType() {
    Book book = getBook();
    String newBookType = ConsoleReader.readNewBookType();
    book.setType(newBookType);
    performUpdate( change: "Book type", book);
}

private static void updateAuthorName() {
    Book book = getBook();
    String newAuthorName = ConsoleReader.readNewAuthorName();
    book.setAuthorName(newAuthorName);
    performUpdate( change: "Book author name", book);
}
```

Class UpdateUtilities

Code

Book store

```java
private static void updatePrice() {
    Book book = getBook();
    String newPrice = ConsoleReader.readNewBookPrice();
    book.setPrice(newPrice);
    performUpdate( change: "Book price", book);
}

private static Book getBook() {
    String bookName = ConsoleReader.readBookName();
    if (CommonUtilities.isBookExist(bookName)) {
        SearchUtilities.searchForBookName(bookName);
        for (int i = 0; i < Book.books.size(); i++) {
            if (Book.books.get(i).getName().equals(bookName)) {
                return Book.books.get(i);
            }
        }
    } else {
        System.out.println("\tInvalid book name");
    }
    return getBook();
}
```

Code

Book store

Class UpdateUtilities

```
    private static void performUpdate(String change, Book book) {
        BookFileHandling.rewriteFile(Book.books);
        SearchUtilities.searchForBookName(book.getName());
        System.out.println("\t" + change + " is updated successfully");
        displayOptions();
    }
}
```

Class UpdateUtilities

Code

Book store

```java
package main.view.common;

import main.classes.Book;
import main.view.administrator.options.AdministratorOptionList;
import main.view.customer.options.CustomerAdvancedOptionList;
import main.view.customer.options.CustomerOptionList;
import main.classes.Customer;

import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;

abstract public class CommonUtilities {
```

Class CommonUtilities

Code

Book store

```java
public static void writeFile(String filePath
        , HashMap<String, ArrayList<String>> list
        , ArrayList<String> keys) {
    try {
        FileWriter myFile = new FileWriter(filePath);
        for (String key : keys) {
            ArrayList<String> bookList = list.get(key);
            myFile.write( str: key + ',');
            for (int i = 0; i < bookList.size(); i++) {
                if (i != bookList.size() - 1) {
                    myFile.write( str: bookList.get(i) + ',');
                } else {
                    myFile.write(bookList.get(i));
                }
            }
            myFile.write( c: '\n');
        }
        myFile.close();
    } catch (IOException error) {
        error.printStackTrace();
    }
}
```

Code

Book store

Class CommonUtilities

```
public static boolean isBookExist(String bookName) {
    for (int i = 0; i < Book.books.size(); i++) {
        if (Book.books.get(i).getName().
                equals(bookName.toLowerCase())) {
            return true;
        }
    }
    return false;
}

public static String getBookPrice(String bookName) {
    for (int i = 0; i < Book.books.size(); i++) {
        if (Book.books.get(i).getName().
                equals(bookName.toLowerCase())) {
            return Book.books.get(i).getPrice();
        }
    }
    return null;
}
```

Class CommonUtilities

Code

Book store

```java
public static String getEmailByCustomer(String customerName) {
    for (int i = 0; i < Customer.customerData.size(); i++) {
        if (Customer.customerData.get(i).getFirstName()
                .equals(customerName)) {
            return Customer.customerData.get(i).getEmailAddress();
        }
    }
    return null;
}

public static double displayBookList(ArrayList<String> books) {
    double totalCost = 0.0;
    for (int i = 0; i < books.size(); i += 2) {
        System.out.println("\nBook name: " + books.get(i) +
                "\nPrice: " + books.get(i + 1) +
                "$" + "\n----------------------------");
        totalCost += Double.parseDouble(books.get(i + 1));
    }
    return totalCost;
}
```

Code

Book store

Class CommonUtilities

```
    public static void returnBackToCustomerMenu() {
        ConsoleReader.makeSpace();
        CustomerOptionList.displayOptionsMenu(
                CustomerOptionList.customerName);
    }

    public static void goToAdvancedMenu() {
        ConsoleReader.makeSpace();
        CustomerAdvancedOptionList.displayMenuOptions(
                CustomerOptionList.customerName);
    }

    public static void returnBackToAdministratorMenu() {
        ConsoleReader.makeSpace();
        AdministratorOptionList.displayOptionsMenu(
                AdministratorOptionList.administratorName);
    }
}
```

Code

Book store

Class CommonUtilities

```java
package main.view.common;

import java.util.Scanner;

abstract public class ConsoleReader {
    public static String getOption() {
        System.out.print("Enter your choice: ");
        Scanner scanner = new Scanner(System.in);
        return scanner.nextLine();
    }

    public static String readFirstName() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter first name: ");
        String firstName = scanner.nextLine();
        if (checkNameValidation(firstName)) {
            return firstName;
        }
        System.out.println("\tName must contain letters only");
        return readFirstName();
    }
```

Class ConsoleReader

Code

Book store

```java
public static String readLastName() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter last name: ");
    String lastName = scanner.nextLine();
    if (checkNameValidation(lastName)) {
        return lastName;
    }
    System.out.println("\tName must contain letters only");
    return readFirstName();
}

private static boolean checkNameValidation(String name) {
    for (int i = 0; i < name.length(); i++) {
        if (!Character.isAlphabetic(name.charAt(i))) {
            return false;
        }
    }
    return true;
}
```

Class ConsoleReader

Code

Book store

```java
public static String readEmailAddress() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter email address: ");
    String emailAddress = scanner.nextLine();
    if (emailAddress.contains("@email.com")) {
        return emailAddress;
    } else {
        System.out.println("\tInvalid email address");
        return readEmailAddress();
    }
}

public static String readPassword() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter password: ");
    return scanner.nextLine();
}

public static String confirmPassword() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter confirm password: ");
    return scanner.nextLine();
}
```

Class ConsoleReader

Code

Book store

```java
public static boolean checkPasswordEquality(String password
        , String confirmPassword) {
    return password.equals(confirmPassword);
}

public static String readPhoneNumber() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter phone number: ");
    String phoneNumber = scanner.nextLine();
    try {
        Integer.parseInt(phoneNumber);
    } catch (NumberFormatException error) {
        System.out.println("\tPhone number must contain numbers only");
        return readPhoneNumber();
    }
    return phoneNumber;
}

public static String readBookName() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter book name: ");
    return scanner.nextLine();
}
```

Class ConsoleReader

Code

Book store

```java
public static String readNewBookName() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter new book name: ");
    return scanner.nextLine();
}

public static String readAuthorName() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter author name: ");
    return scanner.nextLine();
}

public static String readNewAuthorName() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter new author name: ");
    return scanner.nextLine();
}

public static String readBookType() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter book type: ");
    return scanner.nextLine();
}
```

Class ConsoleReader

Code

Book store

```java
    public static String readNewBookType() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter new book type: ");
        return scanner.nextLine();
    }

    public static String readBookPrice() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter book price: ");
        return scanner.nextLine();
    }
    public static String readNewBookPrice() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter new book price: ");
        return scanner.nextLine();
    }

    public static void makeSpace() {
        Scanner scanner = new Scanner(System.in);
        scanner.nextLine();
        for (int i = 1; i <= 10; i++) {
            System.out.println();
        }
    }
}
```

Class ConsoleReader

Code

Book store

```java
package main.view.common;

import main.classes.Book;
import main.view.administrator.options.AdministratorOptionList;
import main.view.customer.options.CustomerOptionList;

import java.util.HashMap;

abstract public class ShowUtilities {
    private enum Options {
        ADVENTURE, HISTORICAL, FANTASY,
        SCIENCE_FICTION, ALL_BOOKS, RETURN_BACK, WRONG
    }

    private static final HashMap<String, Options>
            showOptions = new HashMap<>();

    private static void setOptions() {
        showOptions.put("1", Options.ADVENTURE);
        showOptions.put("2", Options.HISTORICAL);
        showOptions.put("3", Options.FANTASY);
        showOptions.put("4", Options.SCIENCE_FICTION);
        showOptions.put("5", Options.ALL_BOOKS);
        showOptions.put("6", Options.RETURN_BACK);
    }
```

Class ShowUtilities

Code

Book store

```java
private static Options mapper(String option) {
    setOptions();
    if (showOptions.get(option) == null) {
        return Options.WRONG;
    }
    return showOptions.get(option);
}

public static void displayOptionsMenu() {
    displayWelcome();
    System.out.println("\t1- Adventure books");
    System.out.println("\t2- Historical books");
    System.out.println("\t3- Fantasy books");
    System.out.println("\t4- Science fiction books");
    System.out.println("\t5- All books");
    System.out.println("\t6- Return back");
    executeOption(ConsoleReader.getOption());
}
```

Class ShowUtilities

Code

Book store

```
private static void displayWelcome() {
    ConsoleReader.makeSpace();
    if (CustomerOptionList.customerName != null) {
        System.out.println("\t\t\t ***  Welcome " +
                CustomerOptionList.customerName + "  ***");
    } else {
        System.out.println("\t\t\t ***  Welcome " +
                AdministratorOptionList.administratorName + "  ***");
    }
}
```

Code

Book store

Class ShowUtilities

```java
private static void executeOption(String option) {
    switch (mapper(option)) {
        case ADVENTURE -> showBookList(
                getBookType(Options.ADVENTURE));
        case HISTORICAL -> showBookList(
                getBookType(Options.HISTORICAL));
        case FANTASY -> showBookList(
                getBookType(Options.FANTASY));
        case SCIENCE_FICTION -> showBookList(
                getBookType(Options.SCIENCE_FICTION));
        case ALL_BOOKS -> showAllBooks();
        case RETURN_BACK -> {
            if (CustomerOptionList.customerName != null) {
                CommonUtilities
                        .returnBackToCustomerMenu();
            } else {
                CommonUtilities.
                        returnBackToAdministratorMenu();
            }
        }
        default -> {
            System.out.println("\tInvalid option");
            displayOptionsMenu();
        }
    }
```

Class ShowUtilities

Code

Book store

```
private static String getBookType(Options option) {
    switch (option) {
        case ADVENTURE -> {
            return "adventure";
        }
        case FANTASY -> {
            return "fantasy";
        }
        case HISTORICAL -> {
            return "historical";
        }
        case SCIENCE_FICTION -> {
            return "science fiction";
        }
    }
    return null;
}
private static void showBookList(String bookType) {
    for (int i = 0; i < Book.books.size(); i++) {
        if (Book.books.get(i).getType().equals(bookType)) {
            System.out.println(Book.books.get(i).toString());
        }
    }
    returnBack();
}
```

Class ShowUtilities

Code

Book store

```
    private static void showAllBooks() {
        for (int i = 0; i < Book.books.size(); i++) {
            System.out.println(Book.books.get(i).toString());
        }
        returnBack();
    }

    private static void returnBack() {
        if (CustomerOptionList.customerName != null) {
            CommonUtilities.goToAdvancedMenu();
        } else {
            CommonUtilities.returnBackToAdministratorMenu();
        }
    }
}
```

Class ShowUtilities

Code

Book store

```java
package main.view.customer.options;

import data.store.BorrowFileHandling;
import main.view.common.CommonUtilities;
import main.view.common.ConsoleReader;

import java.util.ArrayList;
import java.util.Formatter;
import java.util.HashMap;
import java.util.Objects;

abstract public class BorrowUtilities {
    private enum Options {
        PROCESSED_TO_BORROW, RETURN_BACK, WRONG
    }

    private static final HashMap<String, Options>
            borrowOptions = new HashMap<>();

    private static void setOptions() {
        borrowOptions.put("1", Options.PROCESSED_TO_BORROW);
        borrowOptions.put("2", Options.RETURN_BACK);
    }
```

Code

Book store

Class BorrowUtilities

```java
private static Options mapper(String option) {
    setOptions();
    if (borrowOptions.get(option) == null) {
        return Options.WRONG;
    }
    return borrowOptions.get(option);
}

public static void displayOptionsMenu() {
    System.out.println("\t1- Processed to borrow");
    System.out.println("\t2- Return back");
    executeOption(ConsoleReader.getOption());
}

private static void executeOption(String option) {
    switch (mapper(option)) {
        case PROCESSED_TO_BORROW -> borrowBook();
        case RETURN_BACK -> CommonUtilities.
                returnBackToCustomerMenu();
        default -> {
            System.out.println("\tInvalid option");
            displayOptionsMenu();
        }
    }
}
```

Class BorrowUtilities

Code

Book store

```java
public static void showBorrowCart() {
    String key = BorrowFileHandling.
            getKeyByCustomer(
                    CustomerOptionList.customerName);
    if (key != null) {
        BorrowFileHandling.displayValueOfKey(key);
        BorrowUtilities.displayOptionsMenu();
    } else {
        System.out.println("\tCart is empty");
        CommonUtilities.returnBackToCustomerMenu();
    }
}

private static void borrowBook() {
    KnowBillUtilities.knowBorrowBill();
    String key = BorrowFileHandling.getKeyByCustomer(
            CustomerOptionList.customerName);
    BorrowFileHandling.borrowBooks.remove(key);
    BorrowFileHandling.borrowKeys.remove(key);
    CommonUtilities.writeFile(BorrowFileHandling.borrowFilePath
            , BorrowFileHandling.borrowBooks, BorrowFileHandling.borrowKeys);
    CommonUtilities.returnBackToCustomerMenu();
}
```

Code

Book store

Class BorrowUtilities

```java
public static double displayBooksInCart() {
    String key = BorrowFileHandling.getKeyByCustomer(
            CustomerOptionList.customerName);
    ArrayList<String> books =
            BorrowFileHandling.getValueByKey(key);
    return CommonUtilities.displayBookList(books);
}
```

Code

Book store

Class BorrowUtilities

```java
public static void addToBorrowCart() {
    String bookName = ConsoleReader.readBookName().toLowerCase();
    if (CommonUtilities.isBookExist(bookName)) {
        String key = BorrowFileHandling.getKeyByCustomer(
                CustomerOptionList.customerName);
        if (key != null) {
            if (!isExistBefore(key, bookName)) {
                cartIsFull();
            } else {
                System.out.println("\tThis book has been already added");
            }
        } else {
            addNewBook(CustomerOptionList.customerName, bookName);
            confirmAddingBook();
        }
        CommonUtilities.goToAdvancedMenu();
    } else {
        System.out.println("\tInvalid book name");
        addToBorrowCart();
    }
}
```

Code

Book store

Class BorrowUtilities

```java
private static boolean isExistBefore(String key, String bookName) {
    return BorrowFileHandling.borrowBooks.
            get(key).contains(bookName);
}

private static void addNewBook(String customerName
        , String bookName) {
    ArrayList<String> bookList = new ArrayList<>();
    bookList.add(bookName);
    double price = Double.parseDouble(
            Objects.requireNonNull(CommonUtilities.
                    getBookPrice(bookName))) * 0.1f;
    Formatter formatter = new Formatter();
    formatter.format("%.1f", price);
    bookList.add(formatter.toString());
    String key = CommonUtilities
            .getEmailByCustomer(customerName);
    BorrowFileHandling.borrowBooks.put(key, bookList);
    BorrowFileHandling.borrowKeys.add(key);
    CommonUtilities.writeFile(BorrowFileHandling.borrowFilePath
            , BorrowFileHandling.borrowBooks, BorrowFileHandling.borrowKeys);
}
```

Code

Book store

Class BorrowUtilities

```java
private static boolean isExistBefore(String key, String bookName) {
    return BorrowFileHandling.borrowBooks.
            get(key).contains(bookName);
}

private static void addNewBook(String customerName
        , String bookName) {
    ArrayList<String> bookList = new ArrayList<>();
    bookList.add(bookName);
    double price = Double.parseDouble(
            Objects.requireNonNull(CommonUtilities.
                    getBookPrice(bookName))) * 0.1f;
    Formatter formatter = new Formatter();
    formatter.format("%.1f", price);
    bookList.add(formatter.toString());
    String key = CommonUtilities
            .getEmailByCustomer(customerName);
    BorrowFileHandling.borrowBooks.put(key, bookList);
    BorrowFileHandling.borrowKeys.add(key);
    CommonUtilities.writeFile(BorrowFileHandling.borrowFilePath
            , BorrowFileHandling.borrowBooks, BorrowFileHandling.borrowKeys);
}
```

Code

Book store

Class BorrowUtilities

```
    private static void confirmAddingBook() {
        System.out.println("Added Successfully!");
        System.out.println("The cart is carrying 1 Book");
        System.out.println("**------------------------------------------------**");
    }

    private static void cartIsFull() {
        System.out.println("\tThe cart is carrying already 1 book");
        System.out.println("\tCart is full");
        System.out.println("\tCart can be held only 1 book");
    }
}
```

Code

Book store

Class BorrowUtilities

```java
package main.view.customer.options;

import data.store.CartFileHandling;
import main.view.common.CommonUtilities;
import main.view.common.ConsoleReader;

import java.util.ArrayList;
import java.util.HashMap;

abstract public class BuyUtilities {
    private enum Options {
        PROCESSED_TO_BUY, RETURN_BACK, WRONG
    }

    private static final HashMap<String, Options>
            buyOptions = new HashMap<>();

    private static void setOptions() {
        buyOptions.put("1", Options.PROCESSED_TO_BUY);
        buyOptions.put("2", Options.RETURN_BACK);
    }
```

Code

Book store

Class BuyUtilities

```
private static Options mapper(String option) {
    setOptions();
    if (buyOptions.get(option) == null) {
        return Options.WRONG;
    }
    return buyOptions.get(option);
}

public static void displayOptionsMenu() {
    System.out.println("\t1- Processed to buy");
    System.out.println("\t2- Return back");
    executeOption(ConsoleReader.getOption());
}

private static void executeOption(String option) {
    switch (mapper(option)) {
        case PROCESSED_TO_BUY -> buyBook();
        case RETURN_BACK -> CommonUtilities
                .returnBackToCustomerMenu();
        default -> {
            System.out.println("\tInvalid option");
            displayOptionsMenu();
        }
    }
}
```

Class BuyUtilities

Code

Book store

```
public static void showCart() {
    String key = CartFileHandling.
            getKeyByCustomer(CustomerOptionList.customerName);
    if (key != null) {
        CartFileHandling.displayValueOfKey(key);
        BuyUtilities.displayOptionsMenu();
    } else {
        System.out.println("\tCart is empty");
        CommonUtilities.returnBackToCustomerMenu();
    }
}

private static void buyBook() {
    KnowBillUtilities.knowBuyBill();
    String key = CartFileHandling.getKeyByCustomer(
            CustomerOptionList.customerName);
    CartFileHandling.cartBooks.remove(key);
    CartFileHandling.cartKeys.remove(key);
    CommonUtilities.writeFile(CartFileHandling.cartFilePath
            , CartFileHandling.cartBooks, CartFileHandling.cartKeys);
    CommonUtilities.returnBackToCustomerMenu();
}
```

Code

Book store

Class BuyUtilities

```
public static double displayBooksInCart() {
    String key = CartFileHandling.getKeyByCustomer(
            CustomerOptionList.customerName);
    ArrayList<String> books =
            CartFileHandling.getValueByKey(key);
    return CommonUtilities.displayBookList(books);
}
```

Code

Book store

Class BuyUtilities

```
public static void addToBuyCart() {
    String bookName = ConsoleReader.readBookName().toLowerCase();
    if (CommonUtilities.isBookExist(bookName)) {
        String key = CartFileHandling.getKeyByCustomer(
                CustomerOptionList.customerName);
        if (key != null) {
            int size = sizeOfCart(key);
            if (!isExistBefore(key, bookName)) {
                if (size < 3) {
                    addBookToList(key, bookName);
                    confirmAddingBook(key);
                } else {
                    cartIsFull(key);
                }
            } else {
                System.out.println("\tThis book has been already added");
            }
        } else {
            addNewBook(CustomerOptionList.customerName, bookName);
            confirmAddingBook(CartFileHandling.
                    getKeyByCustomer(CustomerOptionList.customerName));
        }
        CommonUtilities.returnBackToCustomerMenu();
    } else {
        System.out.println("\tInvalid book name");
        addToBuyCart();
    }
}
```

Code

Book store

Class BuyUtilities

```java
public static int sizeOfCart(String key) {
    if (key != null) {
        return (CartFileHandling.getValueByKey(key).size() / 2);
    }
    return 0;
}

private static boolean isExistBefore(String key, String bookName) {
    return CartFileHandling.cartBooks.
            get(key).contains(bookName);
}

private static void addBookToList(String key, String bookName) {
    CartFileHandling.cartBooks.get(key).add(bookName);
    CartFileHandling.cartBooks.get(key).add(CommonUtilities
            .getBookPrice(bookName));
    CommonUtilities.writeFile(CartFileHandling.cartFilePath
            , CartFileHandling.cartBooks, CartFileHandling.cartKeys);
}
```

Class BuyUtilities

Code

Book store

```
private static void addNewBook(String customerName
        , String bookName) {
    ArrayList<String> bookList = new ArrayList<>();
    bookList.add(bookName);
    bookList.add(CommonUtilities.getBookPrice(bookName));
    String key = CommonUtilities
            .getEmailByCustomer(customerName);
    CartFileHandling.cartBooks.put(key, bookList);
    CartFileHandling.cartKeys.add(key);
    CommonUtilities.writeFile(CartFileHandling.cartFilePath
            , CartFileHandling.cartBooks, CartFileHandling.cartKeys);
}

private static void confirmAddingBook(String key) {
    System.out.println("Added Successfully!");
    System.out.println("The cart is carrying: " + sizeOfCart(
            key) + " Books");
    System.out.println("**------------------------------------------------**");
}
```

Code

Book store

Class BuyUtilities

```
    private static void cartIsFull(String key) {
        System.out.println("\tThe cart is carrying: " + sizeOfCart(
                key) + " Books");
        System.out.println("\tCart is full");
        System.out.println("\tCart can be held only 3 books");
    }
}
```

Class BuyUtilities

Code

Book store

```java
package main.view.customer.options;

import java.util.HashMap;

import main.view.common.CommonUtilities;
import main.view.common.ConsoleReader;

abstract public class CustomerAdvancedOptionList {
    private enum Options {
        ADD_TO_BUY_CART, ADD_TO_BORROW_CART,
        RETURN_BACK, WRONG
    }

    private static final HashMap<String, Options> userAdvancedOptions
            = new HashMap<>();

    private static void setOptions() {
        userAdvancedOptions.put("1", Options.ADD_TO_BUY_CART);
        userAdvancedOptions.put("2", Options.ADD_TO_BORROW_CART);
        userAdvancedOptions.put("3", Options.RETURN_BACK);
    }
```

Class
CustomerAdvancedOptionList

Book store

```java
private static Options mapper(String option) {
    setOptions();
    if (userAdvancedOptions.get(option) == null) {
        return Options.WRONG;
    }
    return userAdvancedOptions.get(option);
}

public static void displayMenuOptions(String customerName) {
    System.out.println("\t\t\t ***  Welcome " + customerName + "  ***");
    System.out.println("\t1- Add to buy cart(The cart holds " +
            "only 3 books)");
    System.out.println("\t2- Add to borrow cart(You can't borrow " +
            "more than 1 book)");
    System.out.println("\t3- Return back");
    executeOption(ConsoleReader.getOption());
}
```

Class
CustomerAdvancedOptionList

Code

Book store

```
    private static void executeOption(String option) {
        switch (mapper(option)) {
            case ADD_TO_BUY_CART -> BuyUtilities.addToBuyCart();
            case ADD_TO_BORROW_CART -> BorrowUtilities.addToBorrowCart();
            case RETURN_BACK -> CommonUtilities
                    .returnBackToCustomerMenu();
            default -> {
                System.out.println("\tInvalid Option");
                CommonUtilities.goToAdvancedMenu();
            }
        }
    }
}
```

Class
CustomerAdvancedOptionList

Code

Book store

```java
package main.view.customer.options;

import main.view.common.CommonUtilities;
import main.view.common.ConsoleReader;
import main.view.common.ShowUtilities;
import main.view.menu.Menu;

import java.util.HashMap;

abstract public class CustomerOptionList {
    public static String customerName;

    enum Options {
        SEARCH_BOOK, SHOW_BORROW_CART,
        SHOW_BOOKS, SHOW_BUY_CART, RETURN_BOOK,
        KNOW_BILL, LOGOUT, WRONG
    }

    private static final HashMap<String, Options>
            customerOptions = new HashMap<>();
```

Class
CustomerOptionList

Code

Book
store

```
private static void setMenuOptions() {
    customerOptions.put("1", Options.SHOW_BUY_CART);
    customerOptions.put("2", Options.SHOW_BORROW_CART);
    customerOptions.put("3", Options.SHOW_BOOKS);
    customerOptions.put("4", Options.SEARCH_BOOK);
    customerOptions.put("5", Options.KNOW_BILL);
    customerOptions.put("6", Options.LOGOUT);
}

private static Options mapper(String option) {
    setMenuOptions();
    if (customerOptions.get(option) == null) {
        return Options.WRONG;
    }
    return customerOptions.get(option);
}
```

Class
CustomerOptionList

Code

Book store

```
public static void displayOptionsMenu(String customerName) {
    CustomerOptionList.customerName = customerName;
    System.out.println("\t\t\t ***  Welcome " +
            customerName + "  ***");
    System.out.println("\t1- Show buy cart");
    System.out.println("\t2- Show borrow cart");
    System.out.println("\t3- Show book list");
    System.out.println("\t4- Search for a book");
    System.out.println("\t5- Know the bill");
    System.out.println("\t6- Log out");
    executeOption(ConsoleReader.getOption());
}
```

Code

Book store

Class CustomerOptionList

```
    private static void executeOption(String option) {
        switch (mapper(option)) {
            case SEARCH_BOOK -> SearchUtilities.displaySearchMenu();
            case SHOW_BORROW_CART -> BorrowUtilities
                    .showBorrowCart();
            case SHOW_BOOKS -> ShowUtilities.displayOptionsMenu();
            case SHOW_BUY_CART -> BuyUtilities.showCart();
            case KNOW_BILL -> KnowBillUtilities
                    .displayOptionsMenu();
            case LOGOUT -> {
                ConsoleReader.makeSpace();
                Menu.displayMainView();
            }
            default -> {
                System.out.println("\tInvalid Option");
                CommonUtilities.returnBackToCustomerMenu();
            }
        }
    }
}
```

Class
CustomerOptionList

Code

Book
store

```java
package main.view.customer.options;

import data.store.BorrowFileHandling;
import data.store.CartFileHandling;
import main.view.common.CommonUtilities;
import main.view.common.ConsoleReader;

import java.util.ArrayList;
import java.util.HashMap;

abstract public class KnowBillUtilities {
    private enum Options {
        BOUGHT_BOOKS, BORROWED_BOOK,
        RETURN_BACK, WRONG
    }

    private static final HashMap<String, Options>
            knowOptions = new HashMap<>();

    private static void setOptions() {
        knowOptions.put("1", Options.BOUGHT_BOOKS);
        knowOptions.put("2", Options.BORROWED_BOOK);
        knowOptions.put("3", Options.RETURN_BACK);
    }
```

Class
KnowBillUtilities

Code

Book
store

```
private static Options mapper(String option) {
    setOptions();
    if (knowOptions.get(option) == null) {
        return Options.WRONG;
    }
    return knowOptions.get(option);
}

public static void displayOptionsMenu() {
    ConsoleReader.makeSpace();
    System.out.println("\t\t\t ***  Welcome " +
            CustomerOptionList.customerName + "  ***");
    System.out.println("\t1- Bill of bought books");
    System.out.println("\t2- Bill of borrowed book");
    System.out.println("\t3- Return back");
    executeOption(ConsoleReader.getOption());
}
```

Class
KnowBillUtilities

Code

Book store

```
private static void executeOption(String option) {
    switch (mapper(option)) {
        case BOUGHT_BOOKS -> billOfBoughtBooks();
        case BORROWED_BOOK -> billOfBorrowedBooks();
        case RETURN_BACK -> CommonUtilities
                .returnBackToCustomerMenu();
        default -> {
            System.out.println("\tInvalid option");
            displayOptionsMenu();
        }
    }
}
```

Class
KnowBillUtilities

Code

Book
store

```
private static void billOfBoughtBooks() {
    String key = CartFileHandling.getKeyByCustomer(
            CustomerOptionList.customerName);
    if (key != null) {
        double totalCost = BuyUtilities.displayBooksInCart();
        System.out.println("\t Total: " + totalCost + "$");
        System.out.println("============================" +
                "==================================");
    } else {
        System.out.println("\tNo bill available");
    }
    displayOptionsMenu();
}
```

Class
KnowBillUtilities

Code

Book store

```
private static void billOfBorrowedBooks() {
    String key = BorrowFileHandling.getKeyByCustomer(
            CustomerOptionList.customerName);
    if (key != null) {
        ArrayList<String> book = BorrowFileHandling.
                borrowBooks.get(key);
        for (int i = 0; i < book.size(); i += 2) {
            System.out.println("\nBook name: " + book.get(i) +
                    "\nPrice: " + book.get(i + 1) +
                    "$" + "\n----------------------------");
        }
        System.out.println("\t Total: " + book.get(1) + "$");
        System.out.println("============================" +
                "==================================");
    } else {
        System.out.println("\tNo bill available");
    }
    displayOptionsMenu();
}
```

Class
KnowBillUtilities

Code

Book store

```java
public static void knowBuyBill() {
    beginningOFBill();
    double totalCost = BuyUtilities.displayBooksInCart();
    System.out.println("\t Total: " + totalCost + "$");
    endingOfBill();
}

public static void knowBorrowBill() {
    beginningOFBill();
    double totalCost = BorrowUtilities.displayBooksInCart();
    System.out.println("\t Total: " + totalCost + "$");
    endingOfBill();
}

private static void beginningOFBill() {
    System.out.println("\t\t\t\t**Book Store**");
    System.out.println("Date: 18/5/2023");
    System.out.println("Customer Name: " +
            CustomerOptionList.customerName);
    System.out.println("\n============================" +
            "==================================");
}
```

Class
KnowBillUtilities

Code

Book store

```
    private static void endingOfBill() {
        System.out.println("============================" +
                "====================================");
        System.out.println("\t\t\tThank you for trusting us ^_^");
    }
}
```

Class
KnowBillUtilities

Code

Book
store

```java
package main.view.customer.options;

import main.classes.Book;
import main.view.common.CommonUtilities;
import main.view.common.ConsoleReader;

import java.util.HashMap;

abstract public class SearchUtilities {
    enum Options {
        SEARCH_BOOK, SEARCH_AUTHOR, SEARCH_TYPE,
        SEARCH_PRICE, RETURN_BACK, WRONG
    }

    private static final HashMap<String, Options> searchOptions
            = new HashMap<>();

    private static void setSearchOptions() {
        searchOptions.put("1", Options.SEARCH_BOOK);
        searchOptions.put("2", Options.SEARCH_AUTHOR);
        searchOptions.put("3", Options.SEARCH_TYPE);
        searchOptions.put("4", Options.SEARCH_PRICE);
        searchOptions.put("5", Options.RETURN_BACK);
    }
```

Class
SearchUtilities

Code

Book
store

```java
private static Options searchMapper(String option) {
    setSearchOptions();
    if (searchOptions.get(option) == null) {
        return Options.WRONG;
    }
    return searchOptions.get(option);
}

public static void displaySearchMenu() {
    ConsoleReader.makeSpace();
    System.out.println("\t\t\t ***  Welcome " +
            CustomerOptionList.customerName + "  ***");
    System.out.println("\t1- Search for book name:");
    System.out.println("\t2- Search for author:");
    System.out.println("\t3- Search for type:");
    System.out.println("\t4- Search for price:");
    System.out.println("\t5- Return back:");
    executeSearchOptions(ConsoleReader.getOption());
}
```

Class
SearchUtilities

Code

Book
store

```java
private static void executeSearchOptions(String option) {
    switch (searchMapper(option)) {
        case SEARCH_BOOK -> executeSearch(
                Options.SEARCH_BOOK);
        case SEARCH_AUTHOR -> executeSearch(
                Options.SEARCH_AUTHOR);
        case SEARCH_TYPE -> executeSearch(
                Options.SEARCH_TYPE);
        case SEARCH_PRICE -> executeSearch(
                Options.SEARCH_PRICE);
        case RETURN_BACK -> executeSearch(
                Options.RETURN_BACK);
        default -> {
            System.out.println("\tInvalid Option");
            ConsoleReader.makeSpace();
            displaySearchMenu();
        }
    }
}
```

Class
SearchUtilities

Code

Book
store

```java
private static void executeSearch(Options option) {
    if (option == Options.SEARCH_BOOK) {
        String bookName = ConsoleReader.readBookName().
                toLowerCase();
        searchForBookName(bookName);
    } else if (option == Options.SEARCH_AUTHOR) {
        String authorName = ConsoleReader.readAuthorName().
                toLowerCase();
        searchForAuthorName(authorName);
    } else if (option == Options.SEARCH_TYPE) {
        String bookType = ConsoleReader.readBookType().
                toLowerCase();
        searchForBookType(bookType);
    } else if (option == Options.SEARCH_PRICE) {
        String price = ConsoleReader.readBookPrice();
        searchForPrice(price);
    } else {
        CommonUtilities.returnBackToCustomerMenu();
    }
    CommonUtilities.goToAdvancedMenu();
}
```

Class
SearchUtilities

Code

Book
store

```java
public static void searchForBookName(String bookName) {
    for (int i = 0; i < Book.books.size(); i++) {
        if (Book.books.get(i).getName()
                .contains(bookName)) {
            System.out.println(Book.books.get(i).toString());
        }
    }
}
public static void searchForAuthorName(String authorName) {
    for (int i = 0; i < Book.books.size(); i++) {
        if (Book.books.get(i).getAuthorName()
                .contains(authorName)) {
            System.out.println(Book.books.get(i).toString());
        }
    }
}

public static void searchForBookType(String BookType) {
    for (int i = 0; i < Book.books.size(); i++) {
        if (Book.books.get(i).getType()
                .contains(BookType)) {
            System.out.println(Book.books.get(i).toString());
        }
    }
}
```

Class
SearchUtilities

Code

Book
store

```
    public static void searchForPrice(String price) {
        for (int i = 0; i < Book.books.size(); i++) {
            if (Book.books.get(i).getPrice().equals(price)) {
                System.out.println(Book.books.get(i).toString());
            }
        }
    }
}
```

Code

Book store

Class SearchUtilities

```java
package main.view.menu;

import data.store.BookFileHandling;
import data.store.BorrowFileHandling;
import data.store.CartFileHandling;
import data.store.UserFileHandling;
import main.view.common.ConsoleReader;

abstract public class Menu {
    public static void startApplication() {
        UserFileHandling.readFile();
        CartFileHandling.readFile();
        BorrowFileHandling.readFile();
        BookFileHandling.readFile();
        Menu.displayMainView();
    }

    public static void displayMainView() {
        System.out.println("\t\t\t ***  Welcome to Book Store  ***");
        displayMenuOptions();
        MenuOptions.executeMenuOptions(ConsoleReader.getOption());
    }
```

Class Menu

Code

Book store

```
    private static void displayMenuOptions() {
        System.out.println("\t1- Sign Up");
        System.out.println("\t2- Login");
        System.out.println("\t3- Exit");
    }
}
```

Code

Book store

Class Menu

```java
package main.view.menu;

import main.view.common.ConsoleReader;
import main.view.registeration.LogIn;
import main.view.registeration.SignUp;

import java.util.HashMap;

abstract public class MenuOptions {
    private enum Options {
        SIGNUP, LOGIN, EXIT, WRONG
    }

    private static final HashMap<String, Options>
            menuOptions = new HashMap<>();

    private static void setMenuOptions() {
        menuOptions.put("1", Options.SIGNUP);
        menuOptions.put("2", Options.LOGIN);
        menuOptions.put("3", Options.EXIT);
    }
```

Class MenuOptions

Code

Book store

```java
    private static Options mapper(String option) {
        setMenuOptions();
        if (menuOptions.get(option) == null) {
            return Options.WRONG;
        }
        return menuOptions.get(option);
    }

    public static void executeMenuOptions(String option) {
        switch (mapper(option)) {
            case SIGNUP -> SignUp.displaySignUpMenu();
            case LOGIN -> LogIn.displayLoginMenu();
            case EXIT -> System.exit( status: 0);
            default -> {
                System.out.println("\tInvalid option");
                ConsoleReader.makeSpace();
                Menu.displayMainView();
            }
        }
    }
}
```

Class MenuOptions

Code

Book store

```java
package main.view.menu;

import java.util.HashMap;

abstract public class OptionUtilities {
    protected enum Options {
        ADMINISTRATOR, CUSTOMER,RETURN_BACK, WRONG
    }

    protected static final HashMap<String, Options> menuOptions
            = new HashMap<>();

    protected static void setMenuOptions() {
        menuOptions.put("1", Options.ADMINISTRATOR);
        menuOptions.put("2", Options.CUSTOMER);
        menuOptions.put("3", Options.RETURN_BACK);
    }
    protected static Options mapper(String option) {
        setMenuOptions();
        if (menuOptions.get(option) == null) {
            return Options.WRONG;
        }
        return menuOptions.get(option);
    }
}
```

Class OptionUtilities

Code

Book store

```java
package main.view.registeration;

import main.view.common.ConsoleReader;
import main.view.menu.Menu;
import main.view.menu.OptionUtilities;
import main.view.administrator.options.AdministratorOptionList;
import main.view.customer.options.CustomerOptionList;
import main.classes.Administrator;
import main.classes.Customer;

abstract public class LogIn extends OptionUtilities {
    public static void displayLoginMenu() {
        System.out.println("\t1- Login as administrator");
        System.out.println("\t2- Login as customer");
        System.out.println("\t3- Return back");
        executeLoginInOptions(ConsoleReader.getOption());
    }
```

Class LogIn

Code

Book store

```
private static void executeLoginInOptions(String option) {
    switch (mapper(option)) {
        case ADMINISTRATOR -> loginInAsAdministrator();
        case CUSTOMER -> loginInAsCustomer();
        case RETURN_BACK -> {
            ConsoleReader.makeSpace();
            Menu.displayMainView();
        }
        default -> {
            System.out.println("\tInvalid option");
            ConsoleReader.makeSpace();
            displayLoginMenu();
        }
    }
}
```

Class LogIn

Code

Book store

```
    private static void loginInAsAdministrator() {
        String name = Administrator.emailValidation(
                ConsoleReader.readEmailAddress()
                , Administrator.administratorData);
        if (!name.equals("No name")) {
            ConsoleReader.makeSpace();
            AdministratorOptionList.displayOptionsMenu(name);
        } else {
            Menu.displayMainView();
        }
    }

    private static void loginInAsCustomer() {
        String name = Customer.emailValidation(
                ConsoleReader.readEmailAddress()
                , Customer.customerData);
        ConsoleReader.makeSpace();
        CustomerOptionList.displayOptionsMenu(name);
    }
}
```

Class LogIn

Code

Book store

```java
package main.view.registeration;

import data.store.UserFileHandling;
import main.view.common.ConsoleReader;
import main.view.menu.Menu;
import main.view.menu.OptionUtilities;
import main.classes.Administrator;
import main.classes.Customer;
import main.classes.User;

import java.util.ArrayList;

abstract public class SignUp extends OptionUtilities {
    public static void displaySignUpMenu() {
        System.out.println("\t1- Sign up as administrator");
        System.out.println("\t2- Sign up as customer");
        System.out.println("\t3- Return back");
        executeSignUpOptions(ConsoleReader.getOption());
    }
```

Class SignUp

Code

Book store

```java
private static void executeSignUpOptions(String option) {
    switch (mapper(option)) {
        case ADMINISTRATOR -> SignUpAsAdministrator();
        case CUSTOMER -> SignUpAsCustomer();
        case RETURN_BACK -> {
            ConsoleReader.makeSpace();
            Menu.displayMainView();
        }
        default -> {
            System.out.println("\tInvalid option");
            ConsoleReader.makeSpace();
            displaySignUpMenu();
        }
    }
}

private static void SignUpAsAdministrator() {
    Administrator administrator = new Administrator();
    SignUpForUser(UserFileHandling.newAdministratorFilePath
            , administrator, Administrator.inactiveAdministratorData);
    Administrator.inactiveAdministratorData.add(administrator);
    performSignUp();
}
```

Class SignUp

Code

Book store

```
private static void SignUpAsCustomer() {
    Customer customer = new Customer();
    SignUpForUser(UserFileHandling.customerFilePath
            , customer, Customer.customerData);
    Customer.customerData.add(customer);
    performSignUp();
}

private static void SignUpForUser(
        String filePath, User user, ArrayList<User> userData) {
    ArrayList<String> data = getData(userData);
    UserFileHandling.createUser(user, data);
    if (filePath.equals(UserFileHandling.customerFilePath)) {
        UserFileHandling.writeFile(UserFileHandling.
                customerFilePath, data);
    } else {
        UserFileHandling.writeFile(UserFileHandling.
                newAdministratorFilePath, data);
    }
    ConsoleReader.makeSpace();
}
```

Class SignUp

Code

Book store

```java
private static ArrayList<String> getData(ArrayList<User> userData) {
    ArrayList<String> data = new ArrayList<>();
    data.add(ConsoleReader.readFirstName());
    data.add(ConsoleReader.readLastName());
    data.add(getNewEmailAddress(userData));
    String password = ConsoleReader.readPassword();
    String confirmPassword = ConsoleReader.confirmPassword();
    while (!ConsoleReader.checkPasswordEquality(password,
            confirmPassword)) {
        confirmPassword = ConsoleReader.confirmPassword();
    }
    data.add(password);
    data.add(ConsoleReader.readPhoneNumber());
    return data;
}

private static String getNewEmailAddress(ArrayList<User> userData) {
    String emailAddress = ConsoleReader.readEmailAddress();
    if (!isEmailExistBefore(emailAddress, userData)) {
        return emailAddress;
    } else {
        System.out.println("\t Email address has already taken before");
        return getNewEmailAddress(userData);
    }
}
```

Class SignUp

Code

Book store

```
    private static boolean isEmailExistBefore(
            String emailAddress, ArrayList<User> userData) {
        for (User userDatum : userData) {
            if (userDatum.getEmailAddress().equals(emailAddress)) {
                return true;
            }
        }
        return false;
    }

    private static void performSignUp() { Menu.displayMainView(); }
}
```

Code

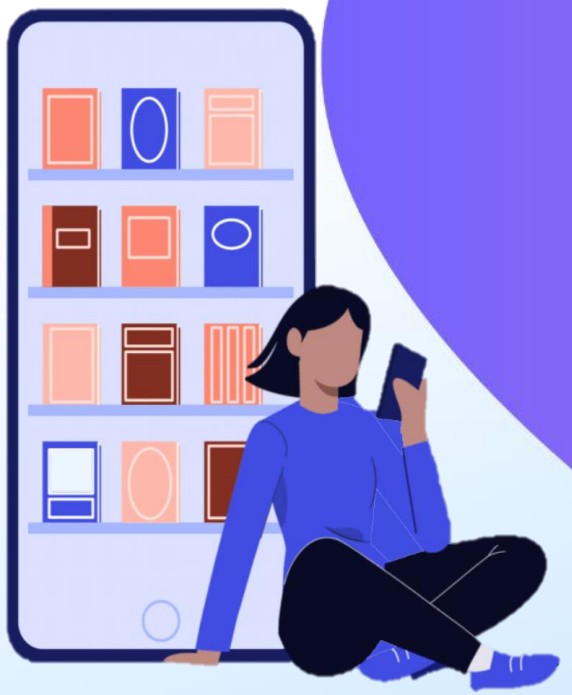
Book store

Class SignUp

Online Book Store

Grop Project

Short w fanila w Lap



Thank You For Listening!