COM102 (Object Oriented Programming) - Practical Skills Assessment 2

Source code – Library Loans

The Items Class:

```
import java.util.List;
import java.util.Scanner;


// -- this items class represents and provides a generic item in the library --
public abstract class Items {

  private String barcode;
  private String artist;
  private String title;
  private int year;
  private String ISBN;

  // -- constructor to initialize item attributes--
  public Items(String barcode, String artist, String title, int year, String ISBN) {
    this.barcode = barcode;
    this.artist = artist;
    this.title = title;
    this.year = year;
    this.ISBN = ISBN;
  }

  public String getBarcode() {
    return barcode;
  }



  // -- getter methods for item attributes --
  public String getArtist() {
    return artist;
  }

  public String getTitle() {
    return title;
  }

  public int getYear() {
    return year;
  }

  public String getISBN() {
    return ISBN;
```

```
}

// -- abstract methods to be implemented by subclasses --

// -- get the type of the item such as "Book" and "Multimedia"--
public abstract String getType();


// -- get the loanable period for the item (in weeks) --
public abstract int getLoanablePeriod();

// -- get the maximum renewal period for the item (in weeks) --
public abstract int getMaxRenewalPeriod();

// -- get the renewal period for the item (in weeks) --
public abstract int getRenewalPeriod();


// -- method to convert barcode to item --
public static Items fromBarcodeToItem(List<Items> items) {
  Scanner scanner = new Scanner(System.in);
  System.out.println("Please provide the Barcode of the item:");
  Items itemLoan = null;
  while (scanner.hasNext()) {
    String inputBarcode = scanner.next();

    // -- iterate through the list of items to find the item with the given barcode --
    for (Items item : items) {
      if (item.getBarcode().equals(inputBarcode)) {
        itemLoan = item;
        break;
      }
    }
    if (itemLoan == null) {
      System.out.println("The Barcode " + inputBarcode + " is incorrect or does not exist");
      System.out.println("Please provide the Barcode of the item:");
      continue;
    }
    break;
  }
  return itemLoan;
}


// -- a toString method to display item information --
@Override
public String toString() {
  return "Barcode: " + barcode +
            "; Artist: " + artist +
            "; Title: " + title +
            "; Year: " + year +
            "; ISBN: " + ISBN;
  }
}
```

The Books Class:

```
import java.time.LocalDate;
import java.util.List;
import java.util.Scanner;

// -- books class representing a book item in the library --
public class Books extends Items {


  // a books constructor to initialize these attributes --
  public Books(String barcode, String artist, String title, int year, String IBSN) {
    super(barcode, artist, title, year, IBSN);
  }


  // -- method to get the type of the item such as "Book" --
  @Override
  public String getType () {
    return "Book";
  }

  // -- method to get the loanable period for the book (in weeks) --
  @Override
  public int getLoanablePeriod() {
    return 5;
  }

  // -- method to get the maximum renewal period for the book (in weeks) --
  @Override
  public int getMaxRenewalPeriod() {
    return 3;
  }


  // -- method to get the renewal period for the book (in weeks) --
  @Override
  public int getRenewalPeriod() {
    return 2;
    }
}
```

The Multimedia Class:

```
import java.time.LocalDate;
import java.util.List;
import java.util.Scanner;

// -- multimedia class representing a multimedia item in the library --
public class Multimedia extends Items {


    // -- this constructor helps to initialize these multimedia attributes that we have --
    public Multimedia(String barcode, String artist, String title, int year, String ISBN) {
        super(barcode, artist, title, year, ISBN);
    }


    // -- method to get the type of the item (Multimedia) --
    @Override
    public String getType() {
        return "Multimedia";
    }

    // -- method to get the loanable period for the multimedia item (in weeks) --
    @Override
    public int getLoanablePeriod() {
        return 5;
    }
   // -- method to get the maximum renewal period for the multimedia item (in weeks) --

    @Override
    public int getMaxRenewalPeriod() {
        return 1;
    }

    // -- method to get the renewal period for the multimedia item (in weeks) --
    @Override
    public int getRenewalPeriod() {
        return 3;
    }
}
```

The User Class:

```
import java.util.List;
import java.util.Scanner;


// -- a users class representing a library user --
public class Users {


  // -- attributes of a user --
  private final String userID;
  private final String firstName;
  private final String lastName;
  private final String email;


  // -- this constructor helps to initialize these attributes --
  public Users(String userID, String firstName, String lastName, String email) {
    this.userID = userID;
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
  }


  // -- these are getter methods for these user attributes --
  public String getUserID() {
    return userID;
  }

  public String getFirstName() {
    return firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public String getEmail() {
    return email;
  }


  // -- method to help convert user ID to user --
  public static Users fromUserIDtoUser(List<Users> users) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Please provide the UserID for the loan\n");
```

```
        Users userLoan = null;
        while (sc.hasNext()) {
            String inputUserID = sc.next();
            for (Users user : users) {
                if (user.getUserID().equals(inputUserID)) {
                    userLoan = user;
                    break;
                }
            }

            // -- so if the user with the given ID is not found, it will prompt the user to try again --
            if (userLoan == null) {
                System.out.println("The UserID " + inputUserID + " is incorrect or does not exist");
                System.out.println("Please provide the UserID for the loan\n");
                continue;
            }
            break;
        }
        return userLoan;
    }
}
```

The Loans Class:

```
import java.time.LocalDate;
import java.util.List;
import java.util.Scanner;


// -- this class represents a loan transaction in the library --
public class Loans {

    private final String barcode;           // -- Barcode of the borrowed item --
    private final String userID;            // -- UserID of the borrowing user --
    private LocalDate issueDate;            // -- Date when the item was borrowed --
    private LocalDate dueDate;              // -- Due date for returning the item --
    private int Renews;                     // --  Number of times the loan has been renewed --



    // -- a constructor which aim is to initialize these loan attributes --
    public Loans(String barcode, String userID, LocalDate issueDate, LocalDate dueDate, int
Renews) {
        this.barcode = barcode;
        this.userID = userID;
        this.issueDate = issueDate;
    this.dueDate = dueDate;
    this.Renews = Renews;
}


// -- getter methods for loan attributes --
public String getBarcode() {
    return barcode;
}

public String getUserID() {
    return userID;
}

public LocalDate getIssueDate() {
    return issueDate;
}

public void setIssueDate(LocalDate issueDate) {
    this.issueDate = issueDate;
}

public LocalDate getDueDate() {
    return dueDate;
}

public void setDueDate(LocalDate dueDate) {
    this.dueDate = dueDate;
}

public int getRenews() {
    return Renews;
}

// -- purpose of this method is to get the item associated with this loan from a list of items --
public Items getItemFromThisLoan(List<Items> items) {
    Items itemLoan = null;


    // -- this is to find the loan's barcode --
    for (Items item : items) {
        if (item.getBarcode().equals(barcode)) {
            itemLoan = item;
            break;
        }
    }

    return itemLoan;
    }

    // -- a method to renew the loan --
    public void renew(List<Items> items) throws RuntimeException {
        Items itemLoan = getItemFromThisLoan(items);


        // -- this is to check if the number of renewals exceeds the maximum renewal period --
        if (this.Renews + 1 > itemLoan.getMaxRenewalPeriod()) {
            throw new RuntimeException("Maximum number of renewals allowed (" + this.Renews +
")");
        }


        // -- this increments the number of renewals and updates the due date --
        this.Renews++;
        this.dueDate = dueDate.plusWeeks(itemLoan.getRenewalPeriod());
    }


    // -- this method returns the loaned item --
    public void returnLoan(List<Items> items) {
        Items itemLoan = getItemFromThisLoan(items);
        LocalDate today = LocalDate.now();

        // -- to see if the item has returned after the due date --
        if (today.isAfter(this.dueDate)) {
            throw new RuntimeException("This item is being returned after its due date (" +
this.dueDate + ")");
        }
    }

    // -- this method converts barcode to loan --
    public static Loans fromBarcodeToLoan(List<Loans> loans) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Please provide the barcode of the loan:");
        Loans itemLoan = null;

        while (scanner.hasNext()) {
            String inputBarcode = scanner.next();

            // -- find the loan with the given barcode --
            for (Loans loan : loans) {
                if (loan.getBarcode().equals(inputBarcode)) {
                    itemLoan = loan;
                    break;
                }
            }
            if (itemLoan == null) {
                System.out.println("The barcode " + inputBarcode + " is incorrect or is not associated
with any loan");
                System.out.println("Please provide the barcode of the loan:");
                continue;
            }

            break;
        }

        return itemLoan;
    }

    // -- a toString method to display the loan info --
    @Override
    public String toString() {
        return "Barcode: " + barcode.toUpperCase() +
                "; UserID: " + userID.toUpperCase() +
                "; Issue date: " + issueDate +
                "; Due date: " + dueDate +
                "; Renewals: " + Renews;
    }
}
```

The Library Management Class:

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


// -- this library management class hands all the library operations and user interface --
public class LibraryManagement {


    // -- file path to get the loans data --
    public static String loans_file = "LOANS.csv";

    // -- a method to load the items from a CSV file --
    public static List<Items> loadItems() {
        final String FILENAME = "ITEMS.csv";
    List<Items> items = new ArrayList<>();

    try {
        File file = new File(FILENAME);
        Scanner scanner = new Scanner(file);


        scanner.nextLine();

        // -- Read the data from file and help create corresponding item objects --
        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();

            String[] info = line.split(",");
            // If it's an empty line, skip it
            if (info.length < 5)
                continue;


            if (info[3].equals("Book"))
                items.add(new Books(info[0], info[1], info[2], Integer.parseInt(info[4]), info[5]));
            else if (info[3].equals("Multimedia"))
                items.add(new Multimedia(info[0], info[1], info[2], Integer.parseInt(info[4]), info[5]));
        }

        scanner.close();
    } catch (IOException e) {
        e.printStackTrace();
    }

    return items;
}


// -- method to load users from the CSV file --
public static List<Users> loadUsers() {
    final String FILENAME = "USERS.csv";
    List<Users> users = new ArrayList<>();

    try {
        File file = new File(FILENAME);
        Scanner scanner = new Scanner(file);


        scanner.nextLine();

        // -- read data from file and create corresponding user objects --
        while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                String[] info = line.split(",");
                // If it's an empty line, skip it
                if (info.length < 3)
                    continue;

                users.add(new Users(info[0], info[1], info[2], info[3]));
            }

            scanner.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return users;
    }


    // Method to load loans from a CSV file
    public static List<Loans> loadLoans() {
        final String FILENAME = loans_file;
        List<Loans> loans = new ArrayList<>();

        try {
            File file = new File(FILENAME);
            Scanner scanner = new Scanner(file);


            if (scanner.hasNextLine())
                scanner.nextLine();

            // -- read data from file and create corresponding loan objects --
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                String[] info = line.split(",");
                // If it's an empty line, skip it
                if (info.length < 4)
                    continue;

                loans.add(new Loans(info[0], info[1], LocalDate.parse(info[2]), LocalDate.parse(info[3]),
Integer.parseInt(info[4])));
            }

            scanner.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return loans;
    }
    // -- another method to save loans to a CSV file --
    public static void saveLoans(List<Loans> loans) {
        try (FileWriter writer = new FileWriter(loans_file)) {
            writer.write("Barcode,UserID,IssueDate,DueDate,Renews\n");
            for (Loans loan : loans) {
                writer.write(loan.getBarcode() + "," + loan.getUserID() + "," + loan.getIssueDate() + "," +
loan.getDueDate() + "," + loan.getRenews() + "\n");
            }
            System.out.println("Loans have been saved to " + loans_file);
        } catch (IOException e) {
            System.err.println("Error while saving loans: " + e.getMessage());
        }
    }

    // -- this is the main menu --
    public static void MainMenu() {
        System.out.println("\nWelcome to the Library Management!\n");
        System.out.println("1 - Issue a loan for a user");
        System.out.println("2 - Renew a loan");
        System.out.println("3 - Return a loan");
        System.out.println("4 - View all active loans");
        System.out.println("5 - View report of the loans");
        System.out.println("6 - Generate library Information");
        System.out.println("0 - Exit from the program\n");
        System.out.println("Enter Your Choice:");
    }

    // -- this is the user options --
    public static void UserOptions(String choice, List<Loans> loans, List<Items> items,
List<Users> users) {
        switch (choice) {
            case "1":
                issueLoan(loans, items, users);
                break;
            case "2":
                renewLoan(loans, items, users);
                break;
            case "3":
                returnLoan(loans, items, users);
                break;
            case "4":
                viewLoans(loans);
                break;
            case "5":
                viewReport(loans, items);
                break;
        case "6":
          itemInfo(items);
          break;
        case "0":
          exit(loans);
          break;
        default:
          System.out.println("Invalid choice. Please try again.");
          break;
      }
    }

    // -- this method issues a new loan --
    public static void issueLoan(List<Loans> loans, List<Items> items, List<Users> users) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter the UserID:");
      String userID = scanner.nextLine();
      System.out.println("Enter the Barcode:");
      String barcode = scanner.nextLine();

      // -- check if user and item exist --
      Users user = null;
      for (Users u : users) {
        if (u.getUserID().equals(userID)) {
          user = u;
          break;
        }
      }

      Items item = null;
      for (Items i : items) {
        if (i.getBarcode().equals(barcode)) {
          item = i;
          break;
        }
      }

      if (user == null || item == null) {
        System.out.println("User or item not found. Loan can't be issued.");
        return;
      }

      LocalDate today = LocalDate.now();
      LocalDate dueDate = today.plusWeeks(item.getLoanablePeriod());
      Loans loan = new Loans(barcode, userID, today, dueDate, 0);
      loans.add(loan);

      System.out.println("Loan issued successfully:");
```

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

// -- this method renews a loan --
public static void renewLoan(List<Loans> loans, List<Items> items, List<Users> users) {
  Scanner scanner = new Scanner(System.in);
  System.out.println("Enter the Barcode of the loan to renew:");
  String barcode = scanner.nextLine();

  Loans loan = null;
  for (Loans l : loans) {
    if (l.getBarcode().equals(barcode)) {
      loan = l;
      break;
    }
  }

  if (loan == null) {
    System.out.println("Loan has not been found. Can't renew.");
    return;
  }

  Items item = null;
  for (Items i : items) {
    if (i.getBarcode().equals(barcode)) {
      item = i;
      break;
    }
  }

  if (item == null) {
    System.out.println("Item has not been found. Can't renew.");
    return;
  }

  try {
    loan.renew(items);
    System.out.println("Loan renewed successfully.");
    System.out.println("New due date: " + loan.getDueDate());
  } catch (RuntimeException e) {
    System.out.println("Failed to renew loan: " + e.getMessage());
  }
}

// -- see all active loans that are present --
public static void viewLoans(List<Loans> loans) {
  System.out.println("Active Loans:");
  for (Loans loan : loans) {
```

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

    // -- return a loan --
    public static void returnLoan(List<Loans> loans, List<Items> items, List<Users> users) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the Barcode of the loan to return:");
        String barcode = scanner.nextLine();

        Loans loanToRemove = null;
        for (Loans loan : loans) {
            if (loan.getBarcode().equals(barcode)) {
                loanToRemove = loan;
                break;
            }
        }

        if (loanToRemove == null) {
            System.out.println("Loan has been not found. Can't return.");
            return;
        }

        try {
            loanToRemove.returnLoan(items);
            loans.remove(loanToRemove);
            System.out.println("Loan returned successful.");
        } catch (RuntimeException e) {
            System.out.println("Failed to return loan: " + e.getMessage());
        }
    }

    // -- method to view a report of these loans --

    public static void viewReport(List<Loans> loans, List<Items> items) {
        int bookLoans = 0;
        int multimediaLoans = 0;
        int renewedLoans = 0;

        for (Loans loan : loans) {
            Items item = loan.getItemFromThisLoan(items);
            if (item instanceof Books) {
                bookLoans++;
            } else if (item instanceof Multimedia) {
                multimediaLoans++;
            }

            if (loan.getRenews() > 0) {
```

```
            renewedLoans++;
        }
    }

    System.out.println("Library Report:");
    System.out.println("Number of Book Loans: " + bookLoans);
    System.out.println("Number of Multimedia Loans: " + multimediaLoans);
    System.out.println("Number of Loans Renewed: " + renewedLoans);
}


// -- method to print information about an item --
public static void itemInfo(List<Items> items) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the Barcode of the item to print information:");
    String barcode = scanner.nextLine();

    Items item = null;
    for (Items i : items) {
        if (i.getBarcode().equals(barcode)) {
            item = i;
            break;
        }
    }

    if (item == null) {
        System.out.println("Item not found.");
    } else {
        System.out.println(item);
    }
}

// -- method to exit the program --
public static void exit(List<Loans> loans) {
    saveLoans(loans);
    System.out.println("Exiting the program.");
}


// -- the main method --
public static void main(String[] args) {
    List<Items> items = loadItems();
    List<Users> users = loadUsers();
    List<Loans> loans = loadLoans();
    Scanner scanner = new Scanner(System.in);
    boolean wantToExit = false;
    while (!wantToExit) {
        MainMenu();
```

```
            String choice = scanner.nextLine();
            UserOptions(choice, loans, items, users);
            if (choice.equals("0")) {
                scanner.close();
                wantToExit = true;
            }
        }
    }
}
```