Assignment 1: Object-Oriented Programming

Library Management System

IMPORTANT: This assignment must be done individually

Read **Section 1** to understand the programming requirements, **Section 2** to understand the programming tasks that you need to carry out and **Section 3** to know how to submit your work.

1. Description

In this assignment, you will create a Java program named `LibMan` for a simple library management system. The purpose of this program is to allow library patrons to check out (borrow) books, librarians to manage the library's collection, and to compute fines for overdue books. You're provided with a `common` package which contains three classes (`DateUtils`, `Genre` and `PatronType`) which you'll use in this assignment. Please do not modify the provided classes.

Library Patrons have the following attributes:

- Patron ID (a unique identifier automatically generated by the system)
- Name
- Date of Birth (DOB)
- Email
- Phone Number
- Patron Type (regular, premium)

The Patron ID is generated automatically by the system using the formula: the letter `P` followed by a unique number, auto-incremented from `1`. The number should be padded with zeros so that it contains at least 3 digits. For instance, the first patron will have the Patron ID of `P001`, the second patron will have `P002`, and so on. The 100th patron will have the ID of `P100`, zero padding is no longer required because the number already has 3 digits.

Library Books have the following attributes:

- ISBN (International Standard Book Number, a unique identifier)
- Title
- Author
- Genre
- Publication Year
- Number of Copies Available

ISBN is a unique book identifier generated automatically by the system. It combines elements like the author's initials, a numerical code for the genre, and the publication year. For example, a book by `John Doe` in the year `2023` with the genre code for programming (e.g., `02`) would have an ISBN like `JD-02-2023`.

Books in the library can be classified into various genres, such as programming, fiction, non-fiction, science fiction, etc. You'll need to use the provided `common.Genre` enum class.

There are two types of patrons in the library system: *regular* patrons and *premium* patrons. Regular patrons can check out a maximum of `3` books at a time, while premium patrons can check out up to `5` books. Patrons are responsible for returning books on time to avoid fines. You are provided with the `common.PatronType` enum class. You should not implement or modify it.

Your program should allow patrons to check out and return books, librarians to add new books to the collection, update book information, and calculate fines for overdue books based on the defined criteria.

2. Task requirements

1. You must create a top-level package named `a1_sid` for the program where `sid` is your student ID.
2. Copy the provided `common` package as a top-level package of your Java project.
3. Specify and implement suitable Java classes for the stated program requirements.
4. Specify and implement a `Book` class that represents a book in the library management system. This class must have the following:
   (a) `generateISBN()` - Generate a unique ISBN for each book.
   (b) `getNumCopiesAvailable()` - Keep track of the number of copies available.
5. Specify and implement a `Patron` class that represents a library patron. This class must have the following:
   (a) `generatePatronID()` – Generate a unique ID for each Patron (e.g. `P001`, `P002`…)
6. The `LibraryTransaction` class represents a transaction in the library, involving a patron, a book, checkout date, due date, return date, and fine amount. This class must have the following:
   (a) The `calculateFine()` method calculates the fine amount based on the difference between the return date and due date, applying the fine rules as specified below.
   ➔ An overdue fine is generated for each book checked out by a patron, depending on the number of days the book is overdue. The fine calculation is as follows:
      - $1.00 per day for books overdue by 1 to 7 days
      - $2.00 per day for books overdue by 8 to 14 days
      - $3.00 per day for books overdue by more than 14 days

(b) The `getDescription()` method generates a detailed description of the transaction, including the patron ID, book ISBN, checkout date, due date, return date (if available), and fine amount (if applicable).

Example:

```
Transaction Details:
    Patron ID: P001
    Book ISBN: A1-01-2021
    Checkout Date: Mon, May 08 2023
    Due Date: Wed, May 10 2023
    Return Date: Sun, May 27 3923
    Fine Amount: $2081933.00
```

7. Specify and implement a `LibraryManager` class that is responsible for managing library book transactions. This class must have the following operations:

(a) Attributes:

- `List<Book> books`: This list holds all the books in the library.
- `List<LibraryTransaction> transactions`: This list contains all the transactions that have occurred in the library.

(b) Methods:

- `addBook(Book book) -` This method adds a book to the library.
- `List<LibraryTransaction> getCheckedOutBooks(Patron patron) -` This method retrieves a list of checked-out books for a specific patron.
- `checkoutBook(Patron patron, Book book, Date checkoutDate, Date dueDate)` - This method allows a patron to check out a book. It first checks if the patron has exceeded their checkout limit based on their patron type. If not, it creates a new `LibraryTransaction`, adds it to the list of transactions, and updates the number of available copies for the book.
- `returnBook(LibraryTransaction transaction, Date returnDate) -` This method allows a patron to return a book, calculates fines (if any), and updates the number of available copies. It sets the return date in the transaction, calculates fines, updates the number of available copies for the book, and prints a success message.
- `List<LibraryTransaction> getOverdueBooks() -` This method returns a list of library transactions representing overdue books that are not returned yet.

Note: To get the current date to use for calculating the number of overdue days, you need to use the `getCurrentDate()` method in the provided `common.DateUtils` class (you should not modify this class).

- `sort()` - This method sorts the list of transactions by patron ID.

8. Specify and implement the `LibraryManProg` class, which is the main program class. This class has a `main` method that performs the following tasks:

    (a) Initialize at least `10` books in the library collection.

    (b) Initialize at least `3` patrons involving both regular and premium patrons.

    (c) Initialize and use to create `5` book transactions

    (d) Print a list of currently checked-out books

    (e) Print list of the overdue books that are not returned yet

    (f) Patron returns the book

    (g) Sort transactions by patron ID

    (h) End the program.

3. Submission requirements

You must submit a single zip file containing your source code to the portal by the due date. The maximum size for your zip file is **1 MB**.

(*) **Note**: your submission must contain only **two** packages (`common` and `s1_sid`). Put all your necessary classes in the `s1_sid` package.

The zip file name must be of the form `a1_sid.zip`, where `sid` is your student identifier (the remaining bits of the file name must not be changed). For example, if your student id is `1512345678` then your zip file must be named `a1_1512345678.zip`.

IMPORTANT: Failure to name the file as shown will result in no marks being given!

NO PLAGIARISM, strict penalty of marks reduction will be applied!