SCHOOL OF SCIENCE, ENGINEERING & TECHNOLOGY

INTE2512 – OBJECT ORIENTED PROGRAMMING

FINAL PROJECT – A VIDEO STORE

Problem Statement

Genie's video store is a small, locally run video rental shop, which dealing mainly in classics movies and TV series. Jason, the store owner, for more than a decade, has used paper-based records to manage the videos items and the loan process. To catch up with the booming internet, Jason has decided to computerize its inventory, and conduct a computer-based rental process to better deal with the increasing number of items and the number of customers the store has to deal with. Furthermore, Jason also decided to include games as another line of rental products for his shop.

As a result, the video rental shop now deals with three basic types of items:

- Old movie records
- DVDs
- Video games

Every item in the shop has the following details:

- ID: a string that has the following format: Ixxx-yyyy
  - 'I' is the capital letter I.
  - 'xxx' is a unique code of 3 digits (e.g. 123)
  - '-' is a single hyphen character
  - 'yyyy' is the year the item was published (e.g. 1980)
  - An example of a valid ID is 'I001-2001'
- Title: captures the whole title of the item (e.g. 'Game of thrones')
- Rental type: Record, DVD, or Game
- Loan type: either 2-day loan or 1-week loan
- Number of copies held in stock (e.g. 1, 2, 3 or more)
- Rental fee (in USD)
- Rental status: either borrowed or available.

Movie records and DVDs can be further divided into several genres including:

- "Action"
- "Horror"
- "Drama"
- "Comedy"

There are three types of customer accounts in this online video store: **Guest account, Regular account and VIP account**.

A customer account has the following attributes.

- ID: has the following format 'Cxxx'
  - 'C' is the capital letter C.
  - 'xxx' is a unique code of 3 digits (e.g. 123)
  - An example of a valid ID is 'C001'
- Name
- Address
- Phone
- List of rentals
- Username
- Password

All the attributes above need to be validated.

There are several rules that differentiate the three types of account.

- A Guest Account only allows the customer to rent maximum 2 items at a time.
- A Regular account or a VIP account can borrow any number of items in the store.
- A Guest customer that borrowed and successfully returned more than 3 items can be promoted to a Regular customer.
- A Regular customer that borrowed and successfully returned more than 5 items can be promoted to a VIP customer.
- A VIP account can accumulate reward points. When a VIP account has equal or more than 100 reward points, the customer can rent 1 item for free. Every rental a VIP customer makes, he will be rewarded 10 points.
- Only regular customers and VIP customers can borrow 2-day items. If a guest customer tried to borrow a 2-day item, your system must print out an appropriate error message.

Analyze, design, and implement a JavaFX store management app for Genie online video rental store. The app must work on Windows, macOS, and Linux.

The program will create new customer accounts and perform rent/return operations on existing items in the system. Your video rental system should have a collection of video items and a list of customers.

The system also has the following features:

- The ability to add, update and delete items from the database of stocked items.
- The ability to add, update customer from the database.
- The ability to increase the number of copies of an existing item (this is done when new stock arrives).
- The ability to read data from and save the data to disk (e.g. text files). This applied for any updates to the customer list and the item list, as described above.
- The ability to rent an item (hence decreasing the number of copies held in stock). It should not be possible to rent an item for which there are no copies held in stock. In this case, the item's rental status should be '**not available**' or '**borrowed**'.
- The ability to return an item (hence increasing the number of copies held in stock).
- The ability to promote a customer (from Guest to Regular or from Regular to VIP).
- The ability to display all items, sorted by titles or IDs.

- The ability to display all customer, sorted by names or IDs.
- The ability to display a group of customers according to their level (e.g. Guest, Regular, or VIP).
- The ability to display all items that have no copies in stock.
- The ability to search for an item that matches a specified title or ID.
    - Searches that match titles should display the information about that item, including title, genre, rental type, and the number of copies available.
- The ability to search for a customer that matches a specified name or ID.
    - Searches that match a customer should display the information about that customer including customer name and customer ID, phone, address.

Any interaction with the system should be via the application's GUI. The system should load the database on startup, and save it before quitting. Several input files including "customers.txt" and "items.txt" are provided. You should be able to specify the name of the database files in the application.

Data should be retrieved and displayed in **no more than 30 seconds**. While the data is being updated, there should be something displayed to let the user know that the app is currently working and how long remaining the user has to wait, e.g. a loading bar, a loading pie, etc.

GENERAL SPECIFICATIONS

1. Your team can decide the structure in your GUI. However, your team name and the store name must be included.

2. Once running, files should be able to be loaded and customers/admin should be able to log-in using their username and password (admin will use a provided account with the username "admin"). Depending on the role, a corresponding menu should be displayed (customer vs admin)

3. Some required functionalities:

    a. Add a new item, update, or delete an existing item
    b. Add new customer or update an existing customer
    c. Promote an existing customer
    d. Rent an item
    e. Return an item
    f. Display all items
    g. Display out-of-stock items
    h. Display all customers
    i. Display group of customers
    j. Search items or customers

4. When the user logs out, all the changes must be updated in the files.

OTHER SPECIFICATIONS

1. **You will need to demonstrate the Object-Oriented programming skills through coding classes, inheritance, function overloading/overriding, and polymorphism in your program.**
2. You need to include the **UML class diagrams** for your system.
3. The app must work on Windows, macOS, and Linux. If your program does not execute at all, you will only be eligible for 50% of your laboratory mark. The teaching staff will NOT be fixing code to make programs compile or for debugging issues during assessment.
4. One team member is responsible for submitting the group's work prior to the deadline. **Late submissions will incur a penalty of 10% per day**. **Submissions which are five days past the deadline will not be accepted and a grade of zero will be given**.
5. Video demonstration: you need to create a short video (less than 10 minutes) consisting of 2 parts. Part 1 explains how you analyze, design, and implement the system. Part 2 shows a working demo of your system. Every team member must be in the presentation.
6. There is a **README** file in the **doc** directory of your submission. It needs to contain contribution information, the GitHub repo URL of your project, and the link to your video demonstration (your video should be uploaded to YouTube - the uploaded time is used to check for late submission. Your video should be).
7. Contribution information:
    a. If all members work well together with equivalently contributed works, then the contribution should be equally divided (e.g. 25% for each member if this is group of 4, or 20% for group of 5, etc..). Otherwise, please discuss within your group to record the actual contributions of all members (e.g. a slightly higher percentage for people with more contribution and vice versa).
    b. Note: the contribution should be considered in overall in terms of **initiative** (help managing the project, contribute excellent ideas), **amount of work, quality of work, and support for other members**, etc.
8. One member submits for the whole team:

    a. In your IntelliJ project, add a new directory named **doc** at the same level as the **src** directory.
    b. Add the system documentation and system design files to this **doc** directory.
    c. In IntelliJ, select **File > Export to Zip File** to export your Intellij project to a zip file.
    d. Submit this zip file to Canvas before the due time.

You can make multiple submissions, but only the last one is graded.