ECE318: Programming Principles for Engineers

Coursework: Individual Project

Application Description

Important deadlines:

1. Submission deadline: 30/11/2025
   - Time: 17:00
   - Submission via Teams
   - What to submit: Code and UML diagrams (Class & Use case diagrams) in a Zip file.
2. Demo dates
   - Tutorial: 3/12/2025
   - Lecture: 4/12/2025

Project Rules

- You have to download the submitted code and diagrams through Teams and will be examined on them.
- Individual submission – the work should be developed solely by you and with no one else!
  1. This is an individual piece of work, hence you will have to design and implement it on your own. No group projects will be accepted.
- Late submission (Penalties):
  - Same day late submission (after submission deadline and before examination):
    - reduction of 10% from overall mark
  - Next day late submission:
    - Reduction of 20% from overall mark
  - 2+ day submission until 5 days:
    - Reduction of 50% from overall mark
  - 5+ day submission:
    - Rejected and total 100% reduction from overall mark
- No show policy:
  - Students that do not appear for the demo with no justification of their absence will be automatically marked with 0.
    - Need to inform the ECE admin office cc'ing Prof. Marnerides in your email and explaining why you will be absent that day.
- Plagiarism:
  - Students submitting copied diagrams or code that they cannot explain will be investigated for plagiarism and will be automatically assigned with a 0 mark. If the case is severe a further case in the departmental council will be raised.
- Notes:
  1. students requesting an extension with the appropriate justification need to ask at least 1 week in advance of the submission deadline.
  2. students with last minute requests for extension without justification will be automatically rejected
     1. unless an unexpected critical circumstance has emerged (e.g., unforeseen family tragedy, illness etc.) that will be evaluated based on the justification provided by the student.

Goal

Design and implement an application for a library manager, allowing administrators to access and manage the books' dataset through an intuitive GUI. The application should enable the admin to view, add, update, and delete books, as well as filter, sort, and generate reports. This project provides hands-on experience in database design and management, including setting up tables and relationships, and implementing CRUD (Create, Read, Update, Delete) operations for direct database interaction. Emphasizing core OOP principles, you'll design flexible classes to represent elements within the dataset, incorporating features like UML diagrams, polymorphism, inheritance, and encapsulation to build a well-structured, modular codebase. Building a GUI is essential for user interaction, enabling data filtering, sorting, and visualization through charts to display trends and insights.

Datasets

- Dataset : `Amazon Books Dataset: Genre, Sub-genre, and Books`
- From Kaggle The project relies on three interconnected .csv files — `Genre_df.csv`, `Sub_Genre_df.csv`, and `Books_df.csv.`

Each file represents a layer of information within the Amazon Books dataset. The relationships between them allow structured data management and querying. Every Main Genre maps to multiple SubGenres and each book has a main genre and a subGenre.

Dataset 1: Genre

- **Title**: This column contains the main genres of books available on Amazon.
- **Number** of Sub-genres: Indicates the count of sub-genres associated with each main genre.
- **URL**: Provides the link to the page on Amazon where books of this genre are listed.

Dataset 2: Sub_Genre

- **Title**: Lists the specific sub-genres within each main genre.
- **Main** Genre: Indicates the overarching genre to which each sub-genre belongs.
- **No.of Books**: Shows the count of books categorized under each sub-genre.
- **URL**: Provides the link to the page on Amazon where books of this sub-genre are listed.

Dataset 3: Books

- **Title**: The title of the book.
- **Author**: Name of the author or publication house.
- **Main Genre**: The main genre the book belongs to.
- **Sub Genre**: The specific sub-genre of the book.
- **Type**: Indicates the format of the book, such as paperback, Kindle, audiobook, or hardcover.
- **Price**: The price of the book.
- **Rating**: The average rating of the book given by users.
- **No of People Rated**: Indicates the count of users who have rated the book.
- **URLs**: Provides the link to the book's page on Amazon for further details and purchase options.

On demonstration

At the examination, you will be required to upload a larger version of the Books_df.csv file (containing more rows but the same columns) into your application and complete the demonstration using that dataset. This ensures that your implementation can efficiently handle scalability and maintain performance consistency across different dataset sizes.

Requirements - what you will be examined on

The exam will be split in two parts, Explanation and Demonstration on two specific scenarios

Explanation - OOP Principles & Java (3 minutes)

- Class diagram
  - Create Class Diagram
  - Relationships between Classes: Describe the associations, dependencies, or generalization relationships between classes.
  - Methods & Attributes: Briefly explain the key methods and attributes in each class. Focus on relevant attributes and methods.
- Use Case diagram
  - Create Use Case Diagram
  - Explanation of Use Case Diagram: Discuss the main actor (Admin) and use cases shown in the diagram.
  - Explain how the actor completes specific tasks through system interactions.
- Answer to 4 questions related to your code.

Demonstration - Implementing details for GUI & Database (4 minutes)

- You are the bookstore administrator. Through the GUI, demonstrate the full functionality of your database integration and system operations.
- For GUI:
  - Have separate pages or tabs to manage Books, Main Genres, and Sub-genres.
  - Functionality Check: Ensure all features (CRUD, Search, Filter, Sorting, Export) operate correctly and without errors.
  - Page Organization & Navigation: Organize content across clearly structured pages with smooth transitions between sections.
  - Error Handling: Display informative messages and guidance when adding, updating, or deleting entries to help users correct input errors or confirm actions.
- For Database / Data Handling:
  - Your database or in-memory collection must support the following operations:
    - For Books:
      - **Presentation**: When a book is selected, display all its details — e.g., Main Genre, Sub-genre, Author, Title, Type, Price, Rating, etc.
      - **CRUD**: Create, Read, Update, and Delete a book record.
      - **Search**: Search for books using keywords that match Title or Author.
      - **Filter**: Filter books first by Main Genre and then by Sub-genre. Allow multiple main genres to be selected simultaneously.
      - **Sorting**: Sort books by Rating and Price (ascending or descending).
      - **Export Reports**: Export the list of books (either full list or filtered/sorted subset) to a PDF file using the same format as the dataset.
    - For Genres and Sub-genres:
      - **Presentation**: For each Main Genre, show its properties and associated Sub-genres with their attributes.
      - **CRUD**: Create, Read, Update, and Delete both main genre and sub-genre entries.
      - **Search**: Search by keyword within genre or sub-genre names.
      - **Calculations**: Compute and display:
        - Total number of books per main/sub-genre
        - Average rating per main/sub-genre
        - Average price per main/sub-genre
      - **Export Reports**: Generate reports (PDF) including the above calculations for any selected main genre or sub-genre.

Marking System

- C. = Creation
- E. = Explanation
- D. = Demonstration

%	Marking System	Requirements
(20)	Class diagram	- **(2.5)** (C&E) : Class Diagram
		- **(5)** (E) : Relationships
		- **(5)** (E) : Methods & Attributes
	Use Case diagram	- **(5)** (C&E) : Use Case Diagram
(20)	OOP Principles	- **(5)** (E) : Encapsulation:
		- **(5)** (E) : Abstraction:
		- **(5)** (E) : Inheritance:
		- **(5)** (E) : Polymorphism:
(35)	Database or Built-in collections	- **(10)**(D) : Presentation & calculation
		- **(10)** (D) : CRUD
		- **(5)** (D) : Search
		- **(5)** (D) : Filter
		- **(5)** (D) : Sorting
		- **(5)** (D) : Export repots
(25)	GUI	- **(15)**(D) : Functionality
		- **(5)** (D) : Page Organization
		- **(2.5)** (D) : Error handling

Technical tips

- **GUI**:
  - **Swing or JavaFX**: Use Swing or JavaFX to create a responsive and interactive interface, ensuring smooth navigation and functionality for the user.
- **Database**:
  - **JDBC**: Recommend JDBC for connecting to the database, enabling CRUD operations and other data management tasks.
  - **Database Choice (MySQL)**: Use MySQL for a robust, server-based database if project scalability is important.
  - **The database is not mandatory** , you can use just built-in collection but probably will crash or the performance will be a detester
- **Class Diagram and Use Case Diagram**:
  - **draw.io**: Use draw.io to create clean and professional diagrams, including class and use case diagrams. Export diagrams as PDF or PNG for easy integration into documentation or reports.
- **Exception Handling**:
  - Use try-catch blocks to handle exceptions and provide user-friendly messages.
  - **GUI Error Handling**:
    - Use JOptionPane (Swing) or ControlsFX (JavaFX) for error dialogs and form validation.
  - **Database Error Handling:**
    - Use HikariCP for efficient JDBC connection pooling and error management.
    - Integrate Log4j or SLF4J for logging database and application errors.

Tips

- Begin by designing the GUI(not implement yet) and then continue with the class diagram.
- For OOP, implement the basic structure of all classes, define the relationships, then add details.
- Have a backup plan to demonstrate requirements if the GUI doesn’t work.
- As you implement features, consider optimizing the app for performance. The dataset is large, make sure the search and sorting functions are efficient to avoid slowdowns.
- Reminder that the database is not mandatory, you can just use built-in Java collections.