INDIVIDUAL ASSIGNMENT (10%) & GROUP PROJECT BRIEFING (10%)

Learning Outcomes:

- CLO3: Design and develop a small-scale software application with UML class diagram for GUI-based applications.
- CLO4: Participate effectively in group work to construct useful computer programs.

Introduction:

This project is divided into two separate assessments:

i. Individual assignment
ii. Group project (4 members per group)

Budget Tracker GUI App Program Briefing:

The budget tracker GUI app helps users manage their finances by tracking income, expenses, and budgets within a user-friendly graphical interface. The key features of this app are:

- **User Authentication**
  - Users can create an account or log in to an existing one to access their personal financial data.
  - Authentication ensures that only authorized users can view and manage their budget information.
- **Dashboard Overview**
  - Upon logging in, users are greeted with a dashboard that provides an overview of their financial status.
  - The dashboard displays key metrics such as total income, total expenses, remaining budget, and savings.
  - The users’ name must also be displayed at the top of the dashboard.
- **Budget Planning**

  - Users can set up budgets for different spending categories, such as groceries, utilities, entertainment, etc.
  - They can specify budget amounts for each category and set budget periods (e.g., monthly, weekly).
  - The app provides visual feedback to indicate whether users are staying within their budget limits.
- **Transaction Management**
  - Users can add, edit, and delete transactions, categorizing them based on income or expense.
  - Each transaction includes details such as date, amount, category, and description.
  - Transactions can be filtered and sorted based on various criteria (e.g., date range, category).

Project Extensions (Optional)

- **Expense Analysis**
  - The app offers visual representations of users' spending habits, such as pie charts or bar graphs.
- **Reminders and Notifications**
  - Users can set up reminders for upcoming bills, payments, or budget deadlines.
  - Notifications alert users about important financial events or when they exceed budget limits.
- **GUI Enhancement**
  - Enhance the user interface with animations, transitions, or custom graphics to improve user experience.

Data persistence using file I/O must be implemented in this project, which means all information must be stored in a text file.

To get from the minimum passing to good grade, all the key features must be implemented. However, to get an excellent grade, the project extension features need to be implemented. Refer to the provided rubric for a more detail grade distribution.

The comprehensive flow and details of the system is as follows:

- From a cold start (when the application is executed), the Login page is displayed.
    - This window should have two options: sign in for an existing user and register for a new user.
    - The username and password information MUST BE stored in a text file. Name the text file appropriately.
    - For an existing user, the system should be able to search from the list of records stored in the text file, whereas for a new user, their information will be appended in the existing file.
- Once the user has successfully logged in, the dashboard overview will be opened and display a message where the program greets the user (eg *"Hi XYZ, Welcome to Budget Tracker"*). Within this dashboard, all the information related to budget stored in a file must be automatically read and displayed. You are highly encouraged to use an ArrayList to perform this operation to achieve a higher grade.
- Within this dashboard overview, the user can choose to perform budget planning or transaction management.
- When the user exits the program, all updated information will be saved (overwritten) in the same text file.

Programming requirements:

- Objects and Classes
  From the login, budget planning, transaction management to dashboard overview, everything must be represented as classes and objects.

- Inheritance
  At least one of the classes must be made abstract, and inherited. The concept of overriding/overloading the parent's methods must be demonstrated.

- Data management
  Use an ArrayList to hold the information on the budget which can be iterated through and searched. When a new budget item is created, the new detail will be added to the ArrayList before storing it on the text file when the application closes.

- GUI
  The designed interface must be easy to use and intuitive, and not too cluttered. The 'look and feel' must be present.

You should apply the OO design principles to organize your classes with extensibility and maintainability in mind. For each of the functionalities that need to be developed, make sure you make a sensible decision as to which class should encapsulate such behavior.

REMINDER:
Referring to the Internet is allowed. **HOWEVER** any attempt to cheat/use directly the model or project from the Internet will lead to **ZERO** point in this project, especially for the Login function where many sample codes are available online (HashMap is not covered in this course, hence any code for login submitted directly using the HashMap will **NOT** be accepted. The code must be rewritten to use either Array or ArrayList.). Likewise, you are not to ask ChatGPT or any other AI tools to provide the total solution for you.

INDIVIDUAL ASSIGNMENT (10%)

- Each group member must write at least ONE class that is required to complete this program. The class must **NOT** be the application class, and must be different from the other group members. If you need to use more than 4 classes, divide the tasks equally among members.
- Each member must also provide the UML diagram for their own class(es). However, only 1 report is required, which contains all the UML diagrams drawn by each member. Inside the report, clearly state which class and UML diagram belong to which member.
- Each member is required to fill out the rubric for individual assignment and submitted individually.

GROUP PROJECT (10%)

- As a group, compile all individual created classes written, and **together**, write the main application class.
- The GUI design, and the overall program will be assessed as part of group project.
- Include the screenshots of your app with elaborated description as part of the report.
- Only one rubric per group submission.

SUBMISSION REQUIREMENTS

- Project report – one submission per group.
    - Brief introduction of the project.
    - Describe the UML class diagrams in detail.
    - Provide screenshots of your app with elaborated description.
- Java Codes – one submission per group.
- Group Project rubric – one submission per group.
- Individual Assignment rubric – submitted individually.

Deadline: Week 14

- Presentation: Tuesday & Thursday during class time (groups will be selected at random).
- Submission: Thursday, 13th June 2024.