Vanier College
Computer Science Department

Programming 2

Final Project

Due Date: By 11:59pm Sunday, May 4, 2025
Presentation Dates: Week of May 5, 2025 (During your lab session)
Evaluation: 30% of final mark
Format: Individual work
Late Submission: none accepted.

General Guidelines When Writing Programs:

Include the following comments at the top of your source codes:

```
// -------------------------------------------------------
// Final Project
// Written by: (include your name and student id)
// For "Programming 2" Section (include number)– Winter 2025
// --------------------------------------------------------
```

- In a comment, give a general explanation of what your program does. As the programming questions get more complex, the explanations will get lengthier.
- Include comments in your program describing the main steps in your program.
- Display a welcome message.
- Display clear prompts for users when you are expecting the user to enter data from the keyboard.
- All output should be displayed with clear messages and in an easy-to-read format.
- End your program with a closing message so that the user knows that the program has terminated.

What to Submit:

- Make **one ZIP** file that includes all of the **.java files** (for source code) **and.txt files** (for I/O operation, readme and documentation).
- Submit the ZIP file via Omnivox.
  - Do not use the RAR (or some other) format!

Project not submitted to the correct location or not in the requested format will not be graded.

Presentation:

Each student needs to have a short (5 min) presentation and demo.

- ✓ Respect the timeline.
- ✓ Submission without presentation is invalid.
- ✓ If the correct solution has been submitted but the student cannot properly present it, points may be deducted, or a zero may be assigned.

Grading Criteria:

- ➢ Format, clarity, completeness, accuracy of output, efficiency, comments, and presentation are counted for grading.

Project Deliverables:

Deliverable	Due Date	Task
Deliverable 1	11:59pm Sunday, April 6, 2025 (Submit via Lea)	**Basic Personal Finance Tracker Structure:** • Create UML diagram with all classes, methods, etc, and relationships documented (.txt or .pdf format). • Implement the basic structure of the Personal Finance Tracker.
Deliverable 2	11:59pm Wednesday, April 16, 2025 (Submit via Lea)	**Additional Functionality and File Operations** • Enhance the Personal Finance Tracker to support additional functionalities. • Implement methods to add, delete, search, and display transactions. • Add support for saving transactions to a text file and loading transactions from a text file. • Test the file operations thoroughly.
Deliverable 3	11:59pm Tuesday, April 29, 2025 (Submit via Lea)	**Exception Handling and User Interface Improvements** • Improve the robustness of the Personal Finance Tracker by adding exception handling. • Implement custom exceptions for better error management. • Refactor the user interface to provide better user experience and error messages. • Test the exception handling and user interface improvements rigorously. • Implement a functional GUI for the Personal Finance Tracker using JavaFX
Deliverable 4	11:59pm Sunday, May 4, 2025 (Submit via Lea)	**Final Polish and Documentation** • Perform final polishing of the Personal Finance Tracker codebase. • Conduct comprehensive testing to ensure all functionalities work as expected. • Write detailed documentation covering usage instructions, code structure, and any other relevant information. • Prepare a README file with setup instructions and usage guidelines.
Deliverable 5	Week of May 5, 2025	• **Project Presentation**

Project Title: Personal Finance Tracker

Description: The Personal Finance Tracker is a Java application that helps users manage their finances through a graphical user interface (GUI). It allows users to add income and expenses, search or delete transactions by description, and track balances over time. Users can save or load financial records to/from a text file. The system includes visual feedback and a spending limit alert system.

This application must demonstrate object-oriented programming (OOP), recursion, exception handling, file input/output, and GUI development using JavaFX.

Minimum Project Requirements:
At a bare minimum, your project must include the following components:

- ➢ *Transaction Class (Abstract Class):*
    - Fields:
        - ✓ description: A brief note on the transaction
        - ✓ amount: Double value representing money
        - ✓ date: A string or LocalDate object representing the date of transaction
    - Abstract Method:
        - ✓ String getSummary(): Returns transaction details

- ➢ *Subclasses: Income and Expense*
    - Inherit from Transaction
    - Expense includes an additional field: category (e.g., Food, Travel, Rent)
    - Override getSummary() to provide detailed descriptions

- ➢ *FinanceManager Class:*
    - Manages a dynamic list of transactions (ArrayList<Transaction>)
    - Required Methods:
        - ✓ void addTransaction(Transaction t)
        - ✓ boolean removeTransaction(String description)
        - ✓ Transaction searchTransactionByDescription(String desc) – implemented using recursion
        - ✓ double getTotalIncome()
        - ✓ double getTotalExpense()

- ➢ *FileHandler Interface:*
    - void saveToFile(String fileName, ArrayList<Transaction>)
    - ArrayList<Transaction> loadFromFile(String fileName)

Note*:* The user should be prompted to enter the file name for reading and writing data.

➢ *Exception Handling:*
- Your implementation should handle possible exceptions to ensure the application does not crash unexpectedly. Both built-in Java exceptions and custom exceptions should be defined and used where appropriate.
  - ✓ Custom Exception Example: TransactionNotFoundException
  - ✓ Handle IOException, NumberFormatException, etc.

Note: The number and types of exception classes will depend on your design and implementation logic.

➢ *MainGUI Class:*
- This class contains the JavaFX GUI that interacts with the user. It provides buttons and input fields for performing various operations and displays data in a form.
- For example:
  - TextFields for entering description, amount, date
  - ComboBox for selecting type (Income or Expense)
  - ComboBox for selecting category (if expense)
  - TableView to display all transactions
  - Pie Chart to show category-wise expense breakdown
  - Buttons for:
    - ✓ Add Transaction
    - ✓ Search by Description
    - ✓ Delete Transaction
    - ✓ Save/Load to/from File
    - ✓ Alert Dialogs for errors or limit warnings

Note: Input validation is required to ensure the user enters valid data, such as numeric values for the amount, etc,.

➢ *Additional Requirement: Custom Feature or Functionality*

-New Feature/Functionality: Students are required to add at least one new feature or functionality to the Personal Finance Tracker, according to their own design. This feature should be distinct from the basic operations already implemented (add, delete, search, display, save/load). The added functionality should demonstrate creative thinking and understanding of the project.

Examples of possible new features include:
- Spending Limit Alerts
  - ✓ Users can set a spending limit per category
  - ✓ If the total expense in any category exceeds the limit, an alert is displayed
- Other Optional Features:
  - ✓ Export to CSV
  - ✓ Filter by date range
  - ✓ Display chart (like bar, pie, etc,.) of spending over time

Note: Ensure your new feature is meaningful and adds value to the system. You may brainstorm other ideas beyond the examples provided.

Note: The additional feature/functionality should be considered in menu/output.

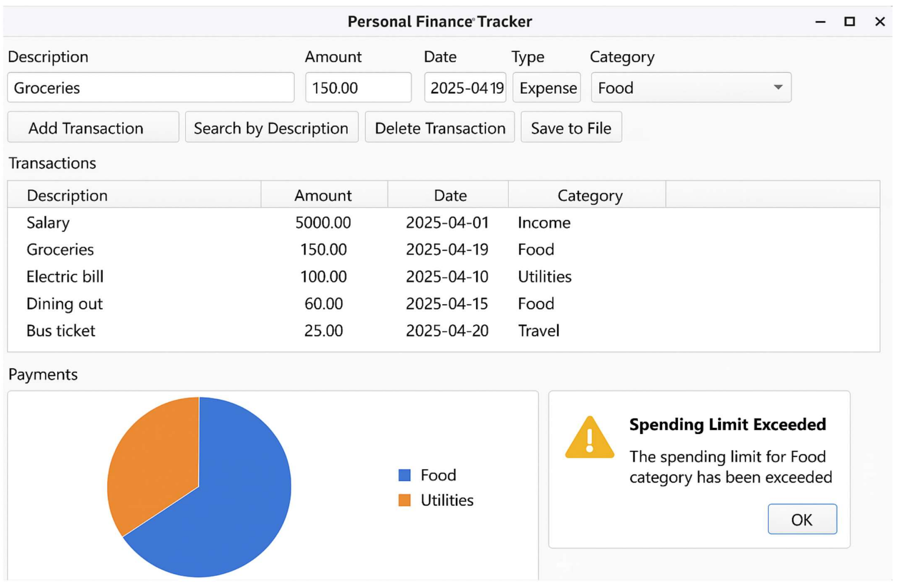
Note: Your classes should have a reasonable set of accessors and mutator methods, constructors, equals and toString methods, whether or not your program uses them.

Note: you can add/define other classes or members including methods and instance variables if you wish.

Note: The format of output is left intentionally vague, allowing considerable design freedom on your part. However, the code should satisfy common standards. Figure 1 and Figure 2 show a simple sample menu in Console format (for your test) and GUI format (for the final deliverable), respectively.

```
----- Personal Finance Tracker -----
1. Add New Transaction
2. Search Transaction by Description
3. Display All Transactions
4. Calculate Total Income / Expense / Balance
5. Save Transactions to File
6. Load Transactions from File
7. Set Spending Limit
8. Exit
```

Figure 1: Sample Console Menu



Description	Amount	Date	Type	Category
Groceries	150.00	2025-04 19	Expense	Food

Transactions

Description	Amount	Date	Category
Salary	5000.00	2025-04-01	Income
Groceries	150.00	2025-04-19	Food
Electric bill	100.00	2025-04-10	Utilities
Dining out	60.00	2025-04-15	Food
Bus ticket	25.00	2025-04-20	Travel

Spending Limit Exceeded

The spending limit for Food category has been exceeded

Figure 2: Sample GUI Form