Expense Tracker for Students

TEAM VARCHAR(4)
SE22UARI165; SE22UARI142; SE22UARI014; SE22UARI141

1. INTRODUCTION

The Finance Tracker App is a desktop application built using Java Swing. It helps users manage their personal finances by tracking incomes and expenses, setting budgets, handling emergency funds, viewing transaction history, and managing profile details.

TECHNICAL DETAILS

1. **Programming Language:**
   Written in Java, using Object-Oriented Programming (OOP) principles.

2. **Libraries Used:**

- javax.swing: For creating the graphical interface.
- java.awt: For arranging the layout and customizing the design.
- java.awt.event: For handling button clicks and other user actions.
- java.util: To store and manage transactions using lists.

GUI DESIGN:

- **Login Screen**: Simple design for entering username and password.

Main App: Organized layout with a list of transactions and buttons for actions.
Class Overview

- **Transaction (Abstract Class):**

Represents a financial transaction.
Contains shared attributes like ID, amount, and date.
Defines an abstract method viewDetails().

- **Income:**

A type of transaction representing money earned.
Adds a specific attribute: source.

- **Expense:**

A type of transaction representing money spent.
Adds a specific attribute: category.

GUI DESIGN:

- **Budget:**

Manages the weekly budget and emergency fund.
Tracks spending, income, and emergency fund usage.

- **TransactionManager:**

Keeps a list of all transactions using an ArrayList.
Provides methods to add and retrieve transactions.

- **FinanceTrackerApp:**

The main application window.
Handles user interactions and connects all features.

- **LoginWindow:**

Provides a login screen for authentication

CORE COMPONENTS

1.1 ***Transaction (Abstract Class)***

A blueprint for all transactions (income and expenses).
Ensures all transactions have common attributes like transactionId, amount, and date.
Has a method viewDetails() that each specific transaction type customizes.

1***.2 Subclasses (Income and Expense)***

- Income: Represents money earned.
- Expense: Represents money spent.

Both inherit from the Transaction class and have their own viewDetails() method to show specific details like category (for expenses) or source (for income).

1.3 Utility Classes

- Budget:

Manages the user's budget and emergency funds.
Handles actions like adding income, tracking spending, and withdrawing emergency funds.

- TransactionManager:

Keeps a list of all transactions using an ArrayList.
Handles adding and managing transactions.

1.4 Main Application (FinanceTrackerApp)

The app's main window built using Java Swing.
Manages the connection between user actions (e.g., clicking buttons) and backend classes like Budget and TransactionManager.

1.5 Login System (LoginWindow)

A separate window for user login.
On successful login, it opens the main Finance Tracker App.

HOW THE GUI WORKS

3.1 Login System (LoginWindow)

- A simple screen with fields for username and password.
- Uses a grid layout to neatly organize inputs and buttons.
- Grants access to the main app after successful login.

3.2 Main Application (FinanceTrackerApp)

Components are arranged in an easy-to-use layout:

- Vertical Stack: All sections are stacked using BoxLayout.
- Button Panel: Buttons (e.g., "Add Income", "View Budget") are arranged in a neat grid.
- Transaction Area: A scrollable area shows all transactions.
  Buttons and Actions

Each button has a specific task (e.g., adding income, viewing budget).
Button clicks trigger backend methods (e.g., addIncome() in the Budget class).

WHY THIS DESIGN IS GREAT

- **Clear Separation:**
  Each part of the app (transactions, budget, GUI) has its own dedicated class.

- **Easy to Expand:**
  New features like "Loan Transactions" can be added by extending the Transaction class.

- **Reusable:**
  Classes like Budget and TransactionManager can be used in other financial apps.

- **Organized:**
  Backend logic (e.g., managing transactions) is separate from the GUI, making the code easy to read and update.

OOP CONCEPTS:

- **Encapsulation:** Attributes like budget, amounts and transactions are private, with public methods to access them.
- **Inheritance:** The abstract Transaction class is extended by Income and Expense classes.
- **Polymorphism:** Different behavior for the viewDetails() method in Income and Expense classes.
- **Abstraction:** The Transaction class provides a blueprint for all transaction types.

OOP CONCEPTS IN THE APP

- **Abstraction**
  The Transaction class hides unnecessary details from users.
  It defines shared features like transactionId and requires subclasses to implement methods like viewDetails().

- **Encapsulation**
  Each class has private attributes (e.g., budgetAmount, transactionId) to protect data.
  Getters and setters provide controlled access to this data.

- **Inheritance**
  Income and Expense inherit from the Transaction class.
  Shared properties and methods are reused, reducing duplicate code.

- **Polymorphism**
  The viewDetails() method works differently for Income and Expense.
  The app treats all transactions as Transaction objects but uses their unique behavior at runtime.

IMPLEMENTATION OVERVIEW

The Finance Tracker App is a personal finance management system designed using [Java Swing] for the user interface and [OOP principles] for backend logic. Here's the breakdown of the implementation:

1. Transaction Class (Abstract)

- This is an abstract class representing a financial transaction. It has the following attributes:
- transactionId: Unique ID for each transaction.
- amount: The value of the transaction.
- date: The date of the transaction.
- The method viewDetails() is abstract and will be implemented by subclasses (Income and Expense) to define how the transaction details are displayed.

JAVA

```
public abstract class Transaction {
    private String transactionId;
    private double amount;
    private String date;

    // Constructor and Getters
    public abstract String viewDetails();
}
```

2. Income Class (Subclass of Transaction)

- Represents an income transaction.
- Adds a new attribute, source, to store where the income comes from (e.g., salary, gift).
- Implements the viewDetails() method to return a string representing the income details.

JAVA

```
class Income extends Transaction {
    private String source;
    @Override
    public String viewDetails() {
        return "Income: $" + getAmount() + " from " + source + " on " + getDate();
    }
}
```

IMPLEMENTATION OVERVIEW

3. Expense Class (Subclass of Transaction)

- Represents an expense transaction.
- Adds a new attribute, category, to specify what the expense is for (e.g., food, rent).
- Implements the viewDetails() method to return a string representing the expense details.

JAVA

```
class Expense extends Transaction {
   private String category;
   @Override
   public String viewDetails() {
      return "Expense: $" + getAmount() + " on " + category + " on " + getDate();
   }
}
```

4. Budget Class

- Manages the user's budget, tracking both the main budget and an emergency fund.

Contains methods to:

- Track spending (trackSpending()).
- Add income (addIncome()).
- Withdraw from emergency funds (withdrawEmergencyFund()).
- Get remaining funds and emergency fund balance.

JAVA

```
class Budget {
   private double budgetAmount;
   private double remainingAmount;
   private double emergencyFund;

   public void trackSpending(double amount) {
      remainingAmount -= amount;
   }
   public void addIncome(double amount) {
      remainingAmount += amount;
   }
}
```

IMPLEMENTATION OVERVIEW

5. TransactionManager Class

- Manages the list of transactions.
- Has a method addTransaction() to add transactions to the list and getTransactions() to retrieve them.

JAVA

```
class TransactionManager {
    private List<Transaction> transactions = new ArrayList<>();

    public void addTransaction(Transaction transaction) {
        transactions.add(transaction);
    }

    public List<Transaction> getTransactions() {
        return transactions;
    }
}
```

IMPLEMENTATION OVERVIEW

6. FinanceTrackerApp (Main Application)

- The main GUI class, extending JFrame, sets up the user interface using *Swing* components.

Buttons:

1. addIncomeBtn: Adds an income transaction.
2. addExpenseBtn: Adds an expense transaction.
3. viewBudgetBtn: Displays the remaining budget.
4. viewEmergencyFundsBtn: Shows the emergency fund balance.
5. withdrawEmergencyFundsBtn: Allows the user to withdraw from emergency funds.
6. profileBtn: Lets the user enter personal details.
7. refreshBtn: Refreshes the transaction display area.
8. The user is prompted to enter the initial budget and emergency fund amount at the start.
9. Each action (like adding income or expenses) is handled via ActionListener methods.

JAVA

```
public class FinanceTrackerApp extends JFrame implements ActionListener {
 private TransactionManager transactionManager = new TransactionManager();
 private Budget budget;

 private JTextArea transactionArea;
 private JButton addIncomeBtn, addExpenseBtn, viewBudgetBtn;

 public FinanceTrackerApp() {
 // Initialize GUI components and layout
 setUpButtons();
 setUpTextArea();
 // Initialize Budget
 double initialBudget = Double.parseDouble(JOptionPane.showInputDialog("Enter initial budget"));
 double emergencyFund = Double.parseDouble(JOptionPane.showInputDialog("Enter emergency fund"));
 budget = new Budget(initialBudget, emergencyFund);
 }

 @Override
 public void actionPerformed(ActionEvent e) {
 if (e.getSource() == addIncomeBtn) handleAddIncome();
 else if (e.getSource() == addExpenseBtn) handleAddExpense();
 }
```

IMPLEMENTATION OVERVIEW

7. LoginWindow Class

- Provides a login screen with username and password fields.
- If credentials match ("admin"/"password123"), the main finance tracker window (FinanceTrackerApp) is launched.

JAVA

```
class LoginWindow extends JFrame {
    public LoginWindow() {
        // Set up login screen UI and event listener for login button
        loginButton.addActionListener(e -> {
            if ("admin".equals(username) && "password123".equals(password)) {
                FinanceTrackerApp app = new FinanceTrackerApp();
                app.setVisible(true);
                dispose();
            }
        });
    }
}
```

8. Running the Application

- The main() method starts by showing the login screen. Upon successful login, it opens the FinanceTrackerApp window, which provides a set of buttons to interact with the finance tracker system.

JAVA

```
public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
        LoginWindow login = new LoginWindow();
        login.setVisible(true);
    });
}
```

FLOW OF THE APPLICATION

1. **Login**: The user logs in with preset credentials.
2. **Main Application**: After login, the FinanceTrackerApp GUI is displayed.
3. **Add Income/Expense**: The user can add income or expense transactions. Each transaction updates the TransactionManager and Budget.
4. **View Budget**: Users can view the remaining budget and emergency funds.
5. **Profile Management**: Users can input and view personal profile information.

KEY FEATURES AND FUNCTIONALITY

- **Income and Expense Tracking**: Users can input their income and expenses, which are stored in a transaction history.
- **Budget Management**: Tracks the remaining amount of the budget and updates it based on expenses and income.
- **Emergency Fund**: Allows users to set aside funds for emergencies and withdraw from them when needed.
- **User Profile:** Stores basic personal information such as name, age, and email.

This implementation is modular, making it easy to extend. For example, new transaction types could be added, or the UI could be enhanced without impacting the core logic

1. LOGIN WINDOW

Purpose: Ensures only authorized users can access the app.

How it works:

1. The user enters their username and password.
2. If the correct login (admin/password123) is entered, the app opens.
3. If the login is incorrect, the user is asked to try again.

2. MAIN FINANCE TRACKER APP

After logging in, the user can perform several actions:

- **Key Features:**

1. Add Income (addIncomeBtn)
2. Purpose: Lets the user record income (like salary or gifts).

- **How it works:**

1. The user enters the income source, amount, and date.
2. A new income transaction is created and added to the history.
3. The remaining budget is updated by adding the income.
4. Add Expense (addExpenseBtn)
5. Purpose: Lets the user record an expense (like rent or food).

- **How it works:**

1. The user enters the expense category, amount, and date.
2. A new expense transaction is created and added to the history.
3. The remaining budget is reduced by the expense.
4. View Budget (viewBudgetBtn)
5. Purpose: Shows how much money is left in the budget.

2. MAIN FINANCE TRACKER APP

How it works:

1. When clicked, the app displays the current remaining budget.
2. View Emergency Funds (viewEmergencyFundsBtn)
3. Purpose: Shows the balance of the emergency fund.

How it works:

1. When clicked, the app displays the emergency fund balance.
2. Withdraw Emergency Funds (withdrawEmergencyFundsBtn)
3. Purpose: Lets the user withdraw money from the emergency fund.

How it works:

1. The user enters the amount to withdraw.
2. If the amount is available, it's moved from the emergency fund to the budget.
3. If the amount is too high, the user is informed.
4. Profile Management (profileBtn and viewProfileBtn)
5. Purpose: Lets the user manage their personal information (name, age, email).

How it works:

1. The user enters their profile information.
2. This information can be saved and viewed later.
3. Refresh Transaction Area (refreshBtn)
4. Purpose: Updates the list of transactions (income and expenses).

How it works:

1. When clicked, the app refreshes the transaction history and displays the most recent transactions.

3. BUDGET AND EMERGENCY FUND MANAGEMENT

Budget Management:

- The user sets an initial budget at the start.
- The budget increases when income is added and decreases when expenses are recorded.
- This helps the user track spending and stay within their budget.

Emergency Fund Management:

- The user can set up an emergency fund.
- The fund balance can be viewed at any time.
- If needed, the user can withdraw from it (as long as there are enough funds).

4. TRANSACTION HISTORY

The app logs all transactions (income and expenses) with details like amount, source/category, and date.
The user can see the full transaction history at any time by refreshing the transaction area.

5. PROFILE MANAGEMENT

The user can enter and view personal information (name, age, email).
This makes the app more personalized and keeps track of the user's details.

User Interaction Flow
Login: The user logs in with their username and password.
Budget Setup: The user sets their initial budget and emergency fund.

Main App:
The user can add income, add expenses, view their budget, or withdraw from the emergency fund.
The transaction area shows the details of all income and expenses.

- Profile Management: The user can update and view their profile.
- Emergency Fund Withdrawals: The user can manage their emergency funds.
- Refresh Transactions: The user can refresh the transaction area to see the most up-to-date list.

Summary of Key Features

- Track Income and Expenses: Users can add income and expenses to monitor their financial flow.
- Budget Management: The app helps users keep track of how much is left in their budget.
- Emergency Fund: Users can set aside money for emergencies and withdraw when needed.
- Profile Management: The user can input and view personal details (name, age, email).
- Transaction History: The app stores a record of all transactions, which can be viewed anytime.