Data Structures Project Report

Cash-flow minimizer

Group no. 3

Team Members:

- Farah Wael Khaled (23P0243)
- Mireille Amir (23P0186)
- Andrea Aziz (23P0379)
- Omar Tamer Mohamed (23P0096)
- Fady Osama (23P0223)

Table of Contents

1.0 Project Overview

The "Cash-flow Minimizer" is a program designed to optimize the settlement of debts among multiple participants by minimizing the number of transactions required. Instead of each individual settling their debts directly with each creditor, the program calculates an efficient set of transactions to resolve all debts collectively. By leveraging data structures and algorithms, the system ensures that the debt settlement process is both streamlined and simplified for all participants.

2.0 Objectives

The primary objectives of the project are:

- To represent debts between participants using a graph.
- To use a greedy algorithm to minimize the number of transactions required to settle all debts.
- To allow the addition of participants dynamically.
- To provide functionality for adding debts, minimizing transactions, and clearing all debts.

3.0 Features

1. **Dynamic Participant Management:**
    - The program allows the dynamic addition of participants by their unique identifiers.
2. **Debt Input:**
    - Users can input debts in the format "Person A owes Person B $X," where A and B are participant IDs.
3. **Debt Minimization:**
    - The program calculates an optimized set of transactions to settle all debts using a greedy algorithm.
4. **Graph Representation:**
    - Debts are stored as a directed graph, where the nodes represent participants and the edges represent debts.

5. **Debt Clearance:**
    - Users can clear all debts and reset the system for fresh input.

4.0 Implementation

1. Data Structure

The project uses a 2D matrix to represent the debt graph. Each element matrix[i][j] represents the amount owed by participant i to participant j.

2. Algorithms

Graph Representation:

- A directed graph is used, where nodes represent participants and edges represent the amount one participant owes another.

Greedy Algorithm for Transaction Minimization:

- Net balances are calculated for each participant by summing up incoming and outgoing debts.
- Participants with a negative balance are considered debtors, while those with a positive balance are creditors.
- Debtors and creditors are matched iteratively to settle the maximum possible amount in each transaction, minimizing the total number of transactions.

3. Code Explanation

Core Functions:

- **addDebt(lender, borrower, amount):** Adds a debt from the borrower to the lender in the graph.
- **minimizeTransactions():** Calculates and displays the optimized transactions using the greedy algorithm.
- **displayDebts():** Displays the current debts.
- **clearDebts():** Resets all debts in the graph.
- **printGraph():** Prints the current state of the debt graph.

Sample Execution Flow

1. The user adds debts using the "Add Participant Debt" option.
2. The system displays the current debt graph.
3. The user selects the "Settle All Debts" option to minimize transactions.
4. The system outputs the optimized transactions required to settle all debts.
5. The user can choose to clear all debts and start over or exit the program.

Example

Input:

- Person 1 owes Person 2 $50.
- Person 2 owes Person 3 $50.

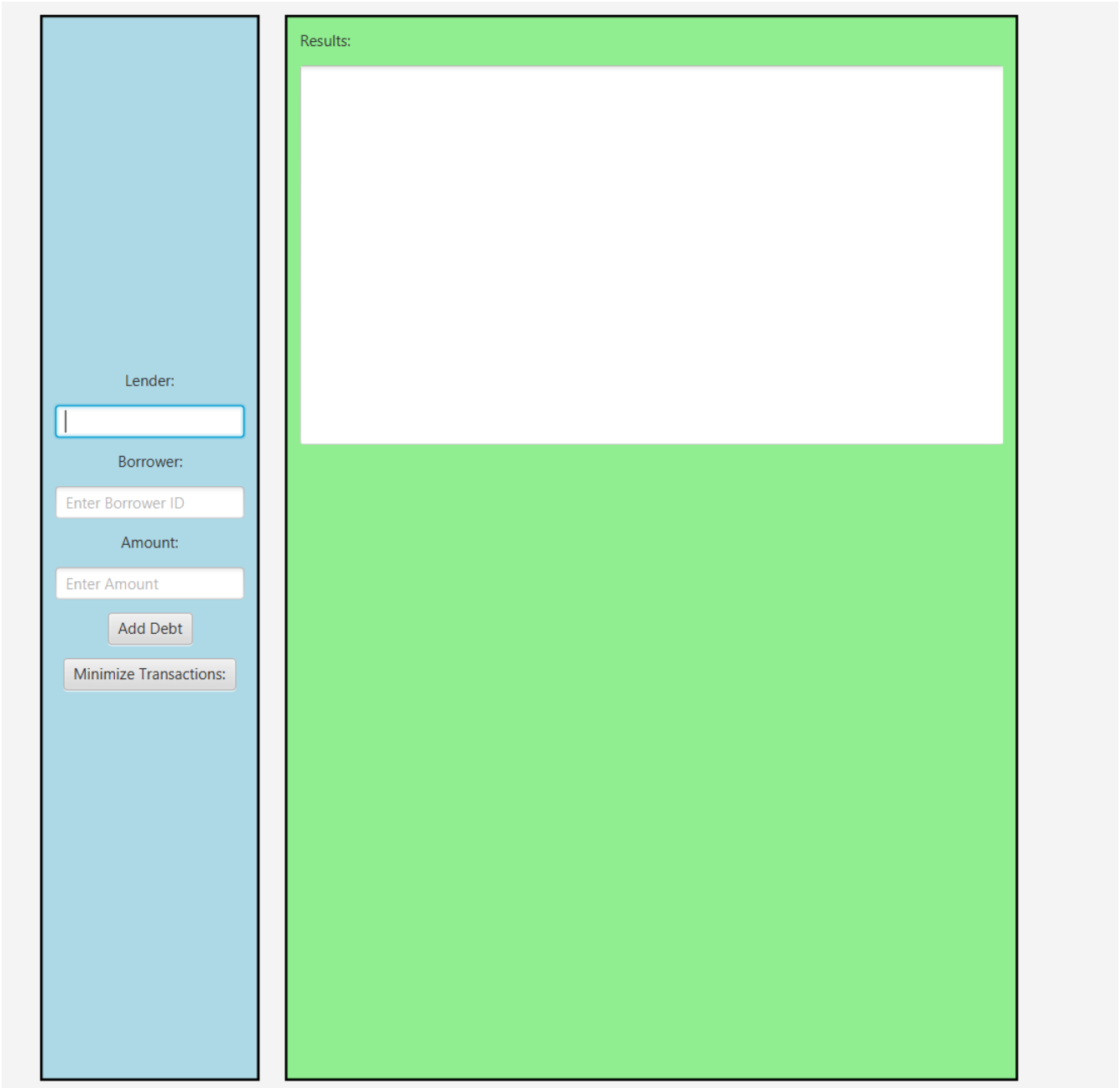
Output:

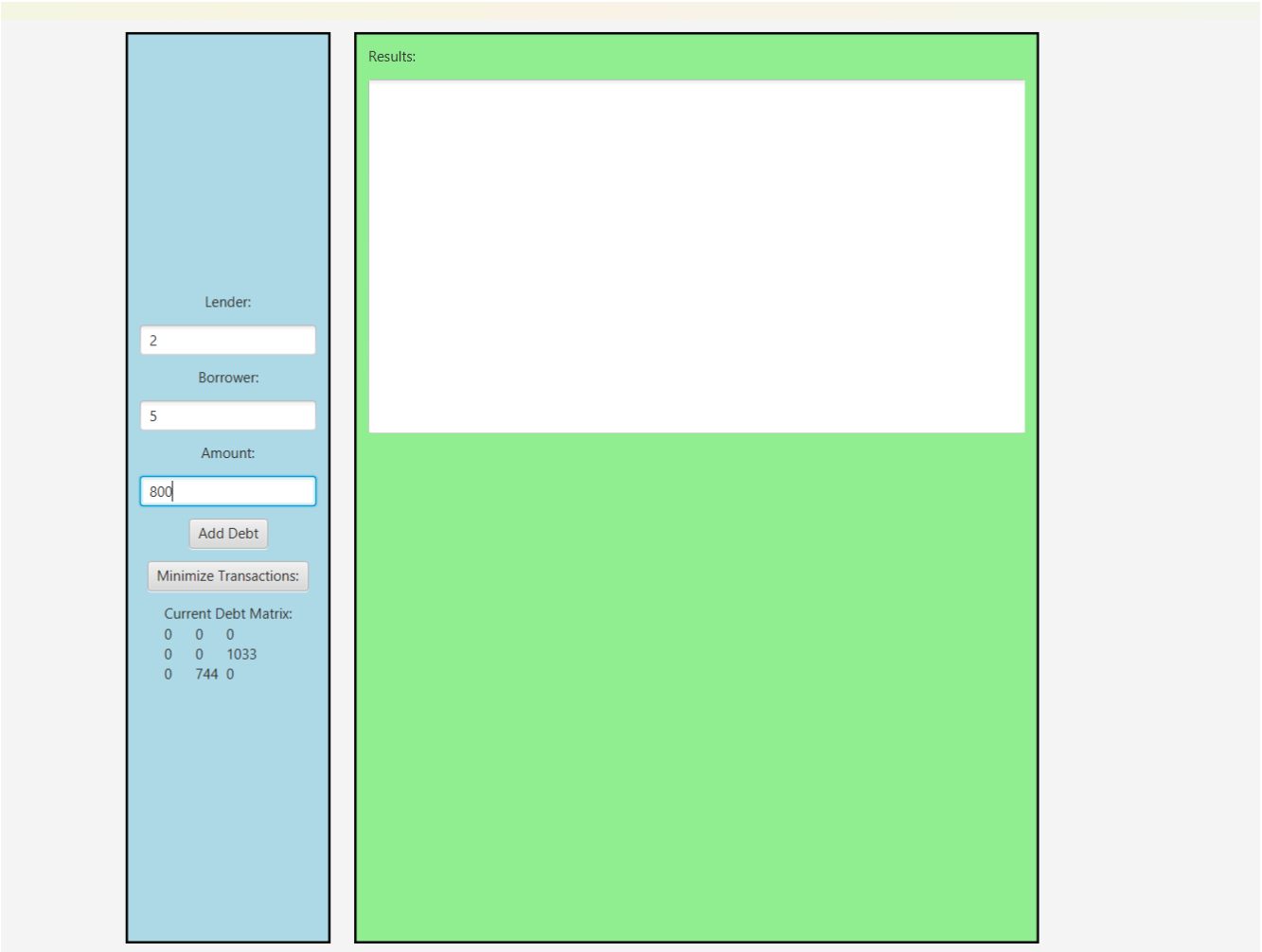
- Optimized Transaction: Person 1 pays Person 3 $50

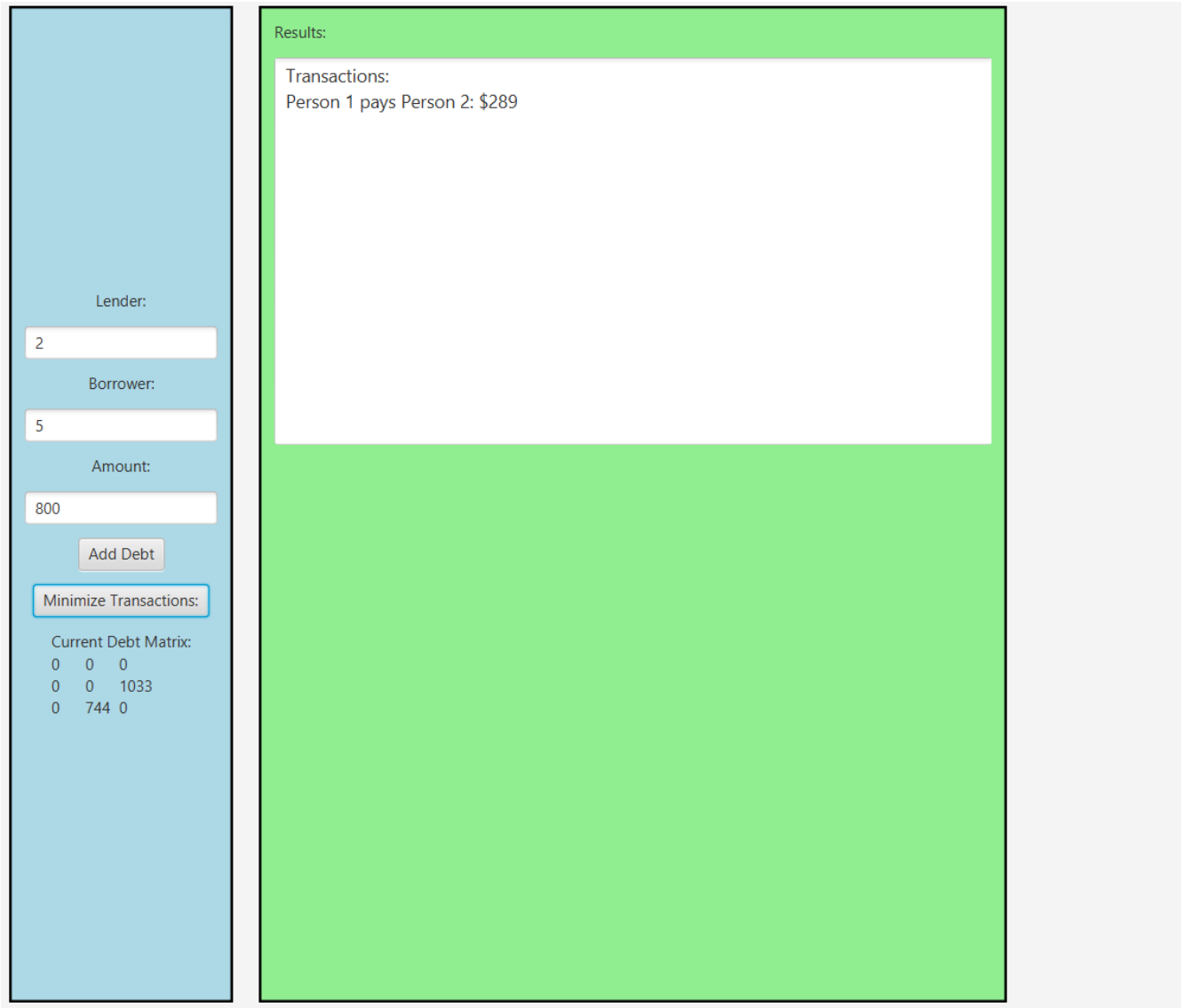
5.0 Bonus Feature: Graphical User Interface (GUI)

As a bonus, a graphical user interface (GUI) was implemented to enhance the project's usability and accessibility. The GUI includes:

1. **Debt Input Form:**
    - A simple interface to input debts by selecting debtors, creditors, and the owed amount.
2. **Transaction Minimization Display:**
    - A clear visualization of the optimized transactions calculated by the program.
3. **Graph Representation:**
    - A graphical depiction of the debt relationships between participants, making it easier to understand the current state of debts.
4. **Debt Clearance:**
    - A one-click option to clear all debts and reset the system.


Lender:
Borrower:
Enter Borrower ID
Amount:
Enter Amount
Add Debt
Minimize Transactions:
Results:


Lender:
2
Borrower:
5
Amount:
800
Add Debt
Minimize Transactions:
Current Debt Matrix:
0 0 0
0 0 1033
0 744 0
Results:


Lender:
2
Borrower:
5
Amount:
800
Add Debt
Minimize Transactions:
Current Debt Matrix:
0 0 0
0 0 1033
0 744 0
Results:
Transactions:
Person 1 pays Person 2: $289

6.0 Conclusion

The "Cash-flow Minimizer" effectively uses data structures and algorithms to solve a real-world problem. By employing a graph to represent debts and a greedy algorithm to minimize transactions, the project demonstrates practical applications of computer science concepts. This tool not only reduces the number of transactions but also simplifies debt settlement among groups.