## George Jimenez, Stephanie Reyes, Juan Meza

## November 11, 2024

CS 3331 – Advanced Object-Oriented Programming – Fall 2024

## Bhanukiran Gurijala

## Project Part: 2

This work was done as a team and completely on our own. We did not share, reproduce, or alter any part of this assignment for any purpose. We did not share code, upload this assignment online in any form, or view/received/modified code written from anyone else. All deliverables were produced entirely on our own. This assignment is part of an academic course at The University of Texas at El Paso and a grade will be assigned for the work we produced.

# **Program Explanation**

In this section, explain the overview of the assignment.

What did you do?

* In this project, we improved by adding new features such reading and processing transactions from files, managing exceptions, handling user input for account creation, producing bank statements, and putting automated tests using JUnit into place. We redesigned code structure, interface usage, design pattern implementation, and expanded upon the framework established in Part 1.

How did you tackle the problem?

* After discussing the needs as a group, our team divided the project into more manageable, smaller jobs. Depending on our unique strengths, each team member took on tasks.

What techniques did you use to solve the problem?

* We used a few strategies, such as interface-based programming to provide flexibility across different classes. For some of the more difficult parts, we also did pair programming, which allowed us to learn from one another and identify any problems early.

Did you break the problem into smaller problems? Explain.

* Together, we broke the issue down into discrete operations including processing files, managing user accounts, processing transactions, and creating bank statements. We were able to concentrate on specific difficulties while making sure that every element would work together seamlessly by tackling each task independently.

# **What did I learn?**

What did you learn as a result of this assignment?

* We gained a deeper understanding of Java's object-oriented capabilities, design patterns, and exception handling during this assignment. We were able to recognize the importance of code reviews and teamwork by working together. Wesaw distinct perspectives that improved our group's comprehension of the project's needs.

How can my solution be improved?

* Employing a more effective data structure for user data storage and retrieval like a hash map for speedier lookups by user ID could further optimize our solution. To further isolate the logic and enhance code readability, a distinct service class tailored for file processing may be developed.

What ideas do you have about another way to solve the problem?

* We talked about a number of different approaches to transaction processing. To encourage a more object-oriented approach to transaction management, one suggestion was to build a command pattern that would encapsulate each transaction type (such as "pay" or "transfer") as an object.

How long did it take me to complete this lab assignment?

* Together, the team worked on the project for about two weeks. Ten hours were spent on a weekly basis and daily check ins while updating and running the code. By cooperating, we were able to divide up the work efficiently and increase our output.

# **Solution Design**

What did you do in this program?

* Key elements of our application include Customer, Bank, Account, Checking, and Savings, Transaction, and BankManager. To provide a clear and maintainable architecture, we created a UML diagram to visually organize these classes and their interactions.

What was your approach to solving this problem?

* We took an object-oriented approach, giving each class a distinct task to perform. While the BankManager class is in charge of high-level banking functions like processing transactions and creating bank statements, the Transaction class, for example, handles transaction data.

What data structures did you use? Why?

* Because lists are easy to iterate over and offer flexibility with dynamic scaling, we chose to utilize them to store users and accounts. We believed lists were adequate for the project scope and would make our code easier to comprehend for this assignment, even if a hash map may provide faster lookup times.

What assumptions, if any, did you make?

* We made the assumptions that every user has a distinct ID and that credit card transactions adhere to predetermined credit limitations that are determined by particular ranges of credit scores. Furthermore, we made the assumption that every transaction activity and CSV column layout would continue to adhere to the specifications.

# **Testing**

How did you test the program?

* We performed white box testing by manually inspecting and testing each part of the code to ensure it worked as intended. This approach allowed us to carefully evaluate the internal logic and verify that each method correctly manipulated account data, processed transactions, and provided the correct feedback.

Did you use black-box, white-box testing, or both? Why?

* We exclusively used white box testing because it allowed us to inspect the internal workings of the system. It was crucial to confirm that the procedures for balance queries, deposits, withdrawals, and transfers operated as planned and interacted with other system components appropriately due to the intricacy of the processes

Did you test the solution enough? How can the testing practices be improved?

* Although we conducted extensive testing, it may be enhanced by automating the tests with a framework for unit testing such as JUnit. As new features are added, this would guarantee consistent findings and improve the efficiency of future testing.

What are the test cases I used?

* Validating user account creation, resolving transaction failures (such as an insufficient balance), processing payments between accounts, and making sure the application throws the proper exceptions for improper activities were some examples of test cases.

Did you break the program and use that to improve it?

* During testing, we discovered a few issues with handling duplicate user names. We resolved these issues by implementing a name-based search feature that distinguishes between users with similar names.

# **Test results**

Describe the results of your tests.

* Our test cases, which included those for creating accounts, generating bank statements, and successful and unsuccessful transactions, all passed in most circumstances. By introducing more understandable exception handling and messages, we were able to fix the minor transaction error handling difficulties we were having.

Include any console outputs showing your results.

Include any text document results of your tests.

A screenshot of a computer

Description automatically generated

# **Code ReviewA screenshot of a phone Description automatically generatedA screenshot of a computer Description automatically generated**

Explain how you conducted a review of your code. Describe how you checked each part of the code review checklist.

* Using a checklist that addressed readability, coding standards, modularity, and exception handling, our team reviewed the code. By having each team member examine each other's code, we were able to identify problems early on and make sure our code followed best practices.

Readability: We looked for logically structured methods and descriptive variable names.  
code Standards: We ensured uniformity in nomenclature and structure by adhering to Java code rules.  
Modularity: To ensure a clear and manageable architecture, we confirmed that every class and method had a defined role.  
Efficiency: To ensure that transactions were handled effectively, we optimized algorithms wherever we could.  
Exception Handling: We examined the application to make sure all methods handled possible exceptions politely and provided users with understandable error messages.