MAY 13, 2024


# ASSIGNMENT NO. 2

## INITIAL IMPLEMENTATION

SOFTWARE ENGINEERING SS'24

KLAURENT MADHI, NIKOLAI MAKLAKOV, ABDUL RAHMAN
Group P

# Table of Contents

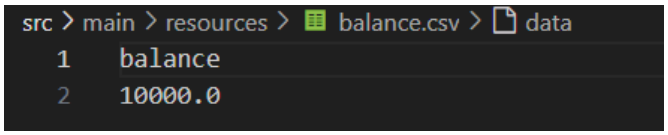
## Introduction

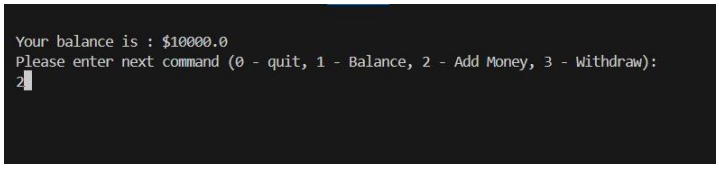
This document details the development of a Simple Banking Application. Designed as a learning tool for those familiar with object-oriented programming concepts, the application simulates basic banking functionalities like deposits, withdrawals, and balance inquiries. Users interact with the system through a user-friendly graphical user interface (GUI) as well as with the command line interface (CLI).

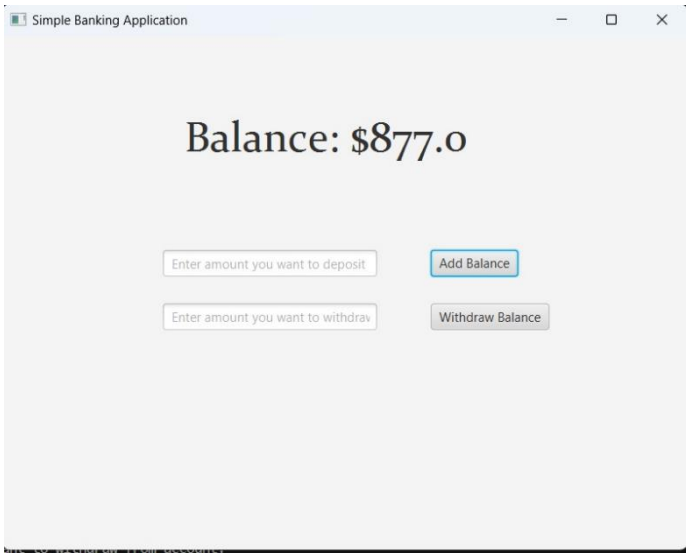
## Completed Features

The following features have been successfully implemented:

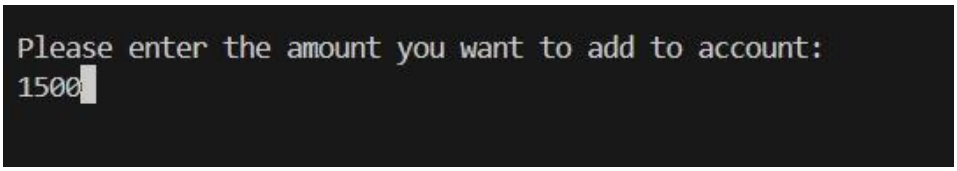
- Initialization: The application retrieves the initial account balance from a data storage.

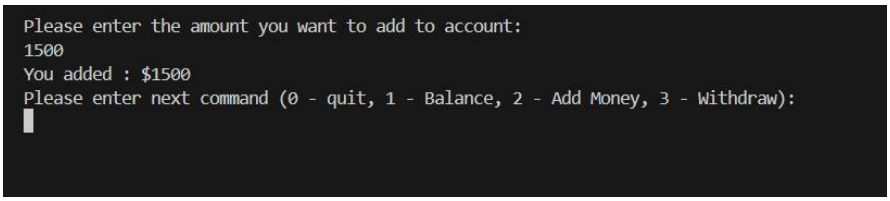


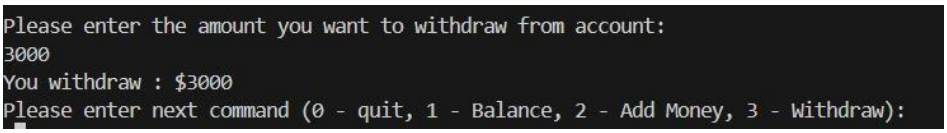




- User Interface (GUI): A simple GUI, includes:
    a. Balance Label: Displays the current account balance.
    b. Text Fields: Allows the users to enter deposit and withdrawal amounts.



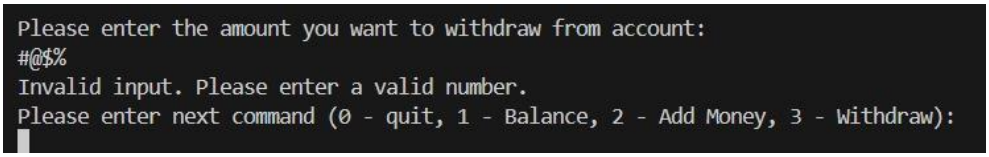



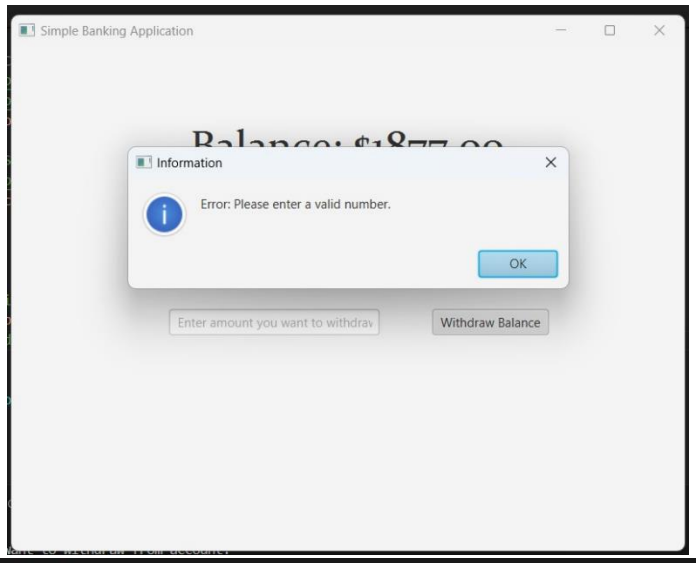

c. Deposit Button: Triggers the deposit functionality upon user interaction.
d. Withdraw Button: Triggers the withdrawal functionality upon user interaction.




e. Deposit Functionality: Users can enter a deposit amount, which is validated for positivity and processed to update the account balance. Error messages are displayed for invalid input.
f. Withdrawal Functionality: Users can enter a withdrawal amount. The system validates the input for positivity and checks for sufficient funds. If insufficient funds exist, an error message is displayed. Otherwise, the withdrawal is processed, and the balance is updated.

- Error Handling: The application handles various error scenarios, including:
  a. Invalid Input: Catches non-numeric entries in the text fields and prompts the user to enter a valid number.

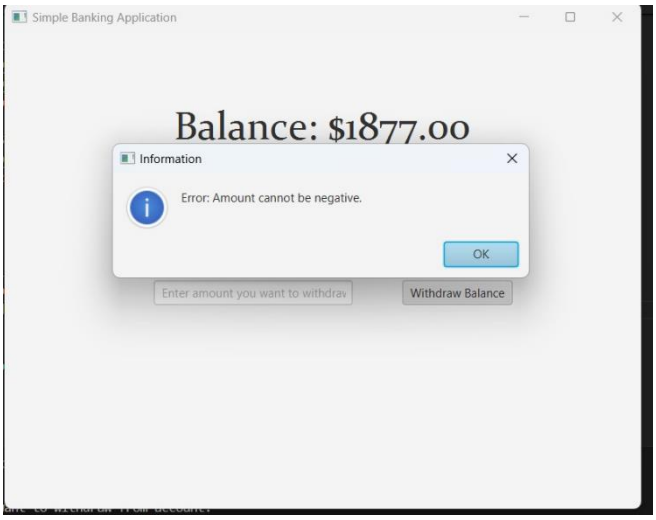




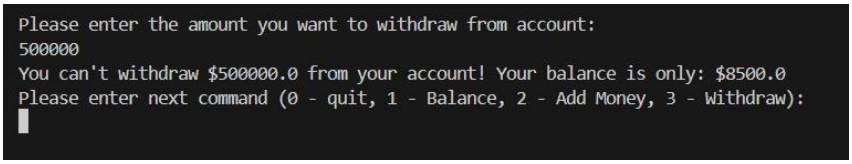

```
Your balance is : $8700.0
Please enter next command (0 - quit, 1 - Balance, 2 - Add Money, 3 - Withdraw):
fsadfasdfasdfsadf
Looks like you pressed a wrong command. Try again in 3 seconds!
```

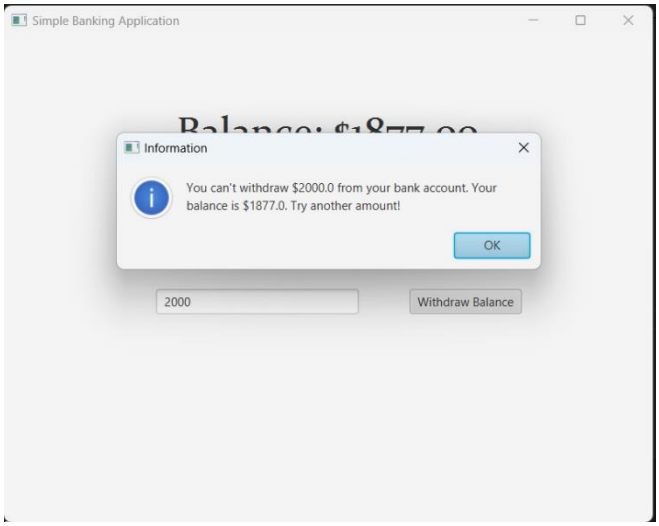
b. Negative Input: Prevents negative deposits or withdrawals and informs the user accordingly.

```
Please enter the amount you want to add to account:
-1000
Error: Amount cannot be negative.
Please enter next command (0 - quit, 1 - Balance, 2 - Add Money, 3 - Withdraw):
```




c. Insufficient Funds: Informs the user when attempting to withdraw more than the available balance.




- Balance Updates: The application updates the account balance after successful deposits or withdrawals and reflects the changes.

## Future Enhancements

While the core functionalities are functional, several enhancements can be considered:

- User Authentication: Implementing login credentials or biometric authentication will significantly improve account security.
- Transaction History: Users benefit from accessing a record of past transactions to better manage their finances. This would require additional data structures and functionalities.
- Multiple Accounts: The ability to create and manage multiple accounts within the system increases user flexibility. This would involve extending the data storage and user interface components.
- Data Persistence: Currently, the balance is retrieved from and stored in a database "*DataStoreSql*". Implementing proper persistence mechanisms would ensure data is saved even after the application is closed.

## Conclusion

The Simple Banking Application effectively demonstrates core Java programming concepts like object-oriented design, user interface development, event handling, data validation, and error management. The current implementation of the code shows the basic functionalities like deposit, withdrawal, and balance updates. Future development can focus on enhancing security, transaction tracking, multi-account support, and data persistence mechanisms.