Team: Sean Chun, Ethan Go
Email: schun@bu.edu

Summary of our Bank Manager Project:

This project was a very challenging task as neither of us has used the JFrame/swing interface before. Learning from the example JFrame classes and various online sources was a daunting task. We were able to figure out the basics of JFrame, but making the project look nice was a difficult task. However, overall we are confident in the back end implementation of the Bank Manager functionality. The manager never interacts with the customer and the customer has access to a variety of basic Bank operations. This includes creating an account, taking out loans and withdrawing money. We've also implemented scenarios for the bank to earn money and to check the overall status of the bank. We wish that our Bank Manager interface looked more appealing but we are satisfied with the overall functionality of our program.

There are some predefined accounts, an example:

Username: Ethan
Password: 0

If the password does not match the one entered before then we will not be allowed to Log In.

Design:

- Objects
    - BankRunnerUI
        - The Main class is here
        - We are setting this as the first mainframe being called when the program is first executed.
    - CustomerUI
        - The CustomerUI is tightly coupled to theCustomer. Its sole purpose is to be the GUI interface. We separated the BackEnd process of the app with FrontEnd through the GUI interfaces.
        - We have a LogIn and Register Page, if you register, it will add a new object to a static array and you can re-login afterwards.
        - The CustomerUI leads to a lot of smaller JFrame classes that has the functionalities a customer has
        - Inside the loans, they have to add collateral (we didn't focus on this) but afterwards they can see a list of the loans they have and click on each individual loan that they have and check each loan's statistics.
        - They have to initialize a checkings and savings since those are not
    - Manager(UI)
        - Future Considerations: Adding a validation system just as the customer
        - Realizing it now, at the end of the assignment we realize that the Manager should have similar behaviors as a Customer. Perhaps making a person parents abstract class for both to inherit could be done.

        - There is only one manager and since he does not have a lot of object composition (in our design and since he's lazy), he doesn't have as much functionality as the customer (who has a lot more buttons to choose and press)
        - The Manager can see a report which would display information from all the Customer's Loan
        - They can also check on each individual's person statistic which contains buttons to each customer and will increase depending on how many customers there is.
    - Bank
        - Contains functions and simple storage of Customers and Loans in ArrayList
        - Additionally, contains the report functions to output the current status of the bank, check customer, and all loans on file
    - Customers
        - Contains a customer initialization (necessary Customer Data)
        - Additionally, every Customer has a unique checking and saving account and a list of Loans created
        - Contains functions that allow for the creation of checking and savings accounts
        - Also contains withdraw and deposit functionality (Done thru the Balance object that is associated with every account)
    - Report
        - Contains a StringBuilder that allows us to construct the report for the Bank Manager
    - Data
        - Customer Data object
        - Name
        - Dob
        - Age
    - Balance
        - Allows us to store the balance of the account
        - Contains getter, subtraction and addition functionality
    - Loans
        - Variables: Value, Interest Rate, length of loan
        - Contains functions that a Loan object should have
            - Getters
            - Value at maturity
    - Account
        - Extends to Checking and Saving
        - Having these separate classes allows us to differentiate between the two accounts and store its balance and the user associated with the account

Gui Design:

- GUI interfaces
    - Bank Runner
    - Manager
    - CustomerUI