Design Document
for the
Front End of Simple Interactive Banking System
(SimBank)

Submitted By:

Tech Testers

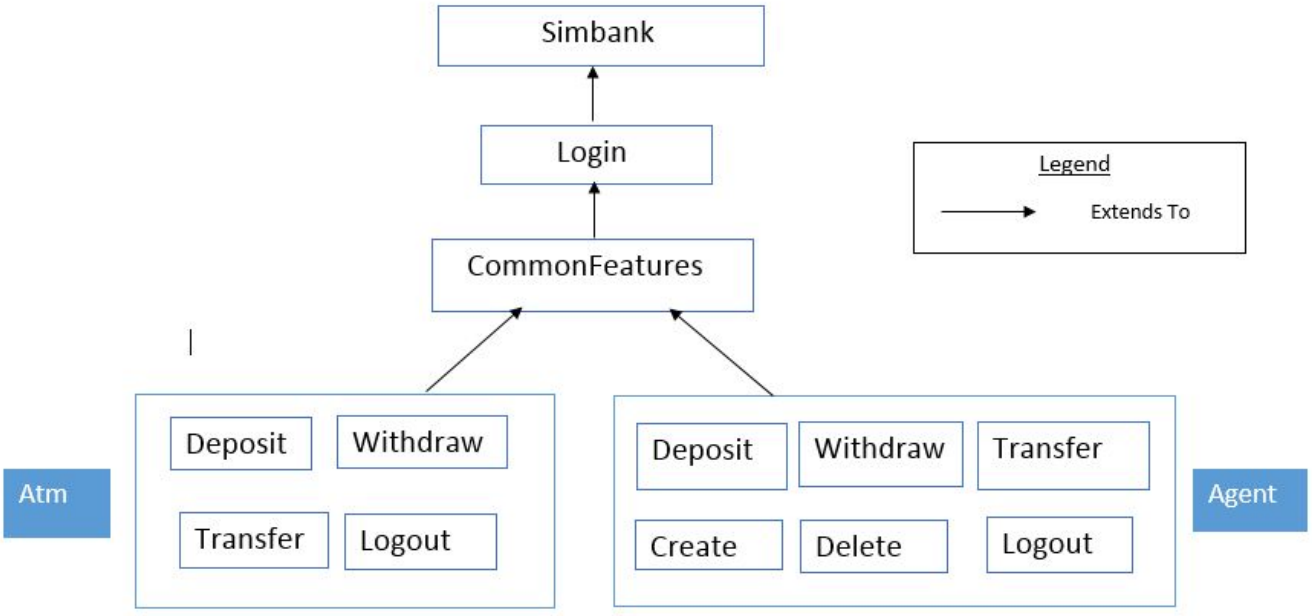
Zainab Bello - 10147946

Wanyu Zhang - 10141510

Jia Yue Sun (Selena) - 10152968

1. OVERVIEW OF THE CLASS HIERARCHY

Diagram showing the class/object hierarchy of Tech Tester's design of the Front End of SimBank



This object hierarchy was designed with the aim to eliminate duplication of code in objects that share common functionality.

The SimBank class is identified as the start of the program, hence it is the most abstract layer. It contains a function call to the next tier - the Login class (extends the SimBank class). This class accepts a valid 'login' command from the user. It retrieves the mode(agent/atm) the user logs into and requests from the user the type of transaction they want to execute. It instantiates the appropriate class depending on the user's input.

The CommonFeatures class extends the Login class and contains methods that define the functionalities shared between various classes in the hierarchy. For the program, the common features shared amongst classes include asking for inputs from the user and checking the validity of the inputs.

The Create, Delete, Deposit, Withdraw, Transfer and Logout classes all extend the CommonFeatures class. The each deal with the start and end of the transaction process requested by the user.

2. DESCRIPTION OF CLASSES AND METHODS

Class SimBank

---

public abstract class **SimBank**

This class reads in a valid accounts file and stores the account information into an arraylist. It prints out a welcome message to the user and ask user to type in “login” to log on to the SimBank system.

Constructor Summary

Constructor and Description
SimBank() **Description:** Creates an instance of the SimBank class with no parameters

Method Detail

Method and Description
public void addArr(String[] array) **Description:** Adds an array of strings containing the transaction summary of a particular banking session to an arraylist **Parameters:** array - an array of strings containing the transaction summary of a particular banking session
private static void getInput() **Description:** Retrieves the ‘login’ command from the user of the system. It also checks if user entered the correct input.
public static void readFile(File filename)

Description: Reads a file and stores data from the valid accounts into an arraylist. **Parameters:** Filename: the name of the valid accounts file
public static void simStart() **Description:** Creates an instance of the Login class marking the beginning of the program.
public static void main(String[] args) **Description:** The starting method of the program. It contains method calls to readFile(File filename) and getInput() **Parameters:** args - the valid accounts file and transaction summary file

Class Login

---

public class **Login**
extends **SimBank**

This class determines what other class to instantiate depending on the transaction mode the user wants to log on to. It makes sure the privileged transaction create and delete are only accepted when logged into agent mode. The method possMode() in this class instantiates the appropriate class in order for the user to continue the transaction process.

Method Detail

Method and Description
private static void possModes(String mode) **Description:** Contains switch statement for each possible transaction after login **Parameters:** mode - the transaction mode the user wants to log into
private static void atmMode() **Description:** Whenever the user logs into the atm mode, this method is called. It ensures privileged transactions (create and delete) aren't allowed for this user.

private static void agentMode() **Description:** Whenever the user logs into the agent mode, this method is called. All transactions are accepted in this mode.
private static String acceptNewTrans() **Description:** Asks the user what transaction they would like to carry out in the current session **Returns:** The user's decision on what transaction they would like to carry out
public static void moreTransacs() **Description:** When a logout transaction is successful, this method asks the user for the following possible transaction. It contains a method call to acceptNewTrans()
public static void loginStart() **Description:** This is the starting method for the Login class. It contains method calls to agentMode() and atmMode()

Class Create

---

class **Create**
extends **CommonFeatures**

This class contains a method call to askUser() in the CommonFeatures class. It prompts the user to input the account number and account name. The checking method for account number and account name are also located in CommonFeatures. It makes sure to write every successful transaction into a transaction summary message in proper format.

Method Detail

Method and Description
private void createAccount (String name, String number) **Description:** Stores the transaction messages of the form CC AAA BBB NNN MMM in the arraylist storing the transaction summary of the current banking session – CC is a two character transaction code in the form: *CR-create* – AAA is the first (to) account number – BBB represents the (from) account number in the form: *00000000*

– NNN is the amount, in cents in the form: *000* – MMM represents the field for the account name **Parameters:** name - the account name of the user currently logged on to the system number - the account number of the user currently logged on to the system
public void createStart() **Description:** Deals with the process of creating a new account for the user.

Class Delete

---

class **Delete**
extends **CommonFeatures**

This class contains a method call to askUser() in the CommonFeatures class. It prompts the user to input the account number and account name. The checking method for account number and account name are also located in CommonFeatures. It makes sure to write every successful transaction into a transaction summary message in proper format.

Method Detail

Method and Description
public void deleteAccount(String number, String name) **Description:** Stores the transaction messages of the form CC AAA BBB NNN MMM in the arraylist storing the transaction summary of the current banking session – CC is a two character transaction code in the form: *DL-delete* – AAA is the first (to) account number – BBB represents the (from) account number in the form: *00000000* – NNN is the amount, in cents in the form: *000* – MMM represents the field for the account name **Parameters:** name - the account name of the user currently logged on to the system number - the account number of the user currently logged on to the system
public void deleteStart() **Description:** Deals with the process of deleting a user's account.

Class CommonFeatures

---

public abstract class **CommonFeatures**
extends **SimBank**

This class contains all of the methods that implement the common functionalities existing within the classes that extend it. The purpose of this class is to minimize code duplication.

Method and Description
public static String askUser(String inUser) **Description:** Retrieves input from the user. **Parameter:** inUser - input from the user. This could be one of the following three: the user account number, name and the amount value to be in the transaction **Returns:** The method returns either the user's account number, name or the amount value to be used in the transaction process
public Boolean checkAccountNo(String inputNo) **Description:** Checks that the entered account number is valid in the sense that it meets all the requirements of a valid account number **Parameter:** inputNo **-** the account number entered by the user **Returns:** True if the account number is valid and false otherwise.
public Boolean validateAccountNo(String inputNo, String mode) **Description:** Checks if the input account number exists within the valid accounts file **Parameters:** inputNo - the account number mode - the transaction mode of the system the user is currently in **Returns:** True if the account number exists in the valid accounts file and false otherwise.
public Boolean checkAccountName(String userName) **Description:** Validates the account name entered by the user **Parameter:**

userName - the user's account name **Returns:** True if the account name is valid and false otherwise.
public Boolean checkAmount(String amount) **Description:** Validates the amount entered by the user. **Parameter:** amount **-** the amount entered by user to be used for transaction **Returns:** True if the amount value is valid and false otherwise.
public Boolean checkAccumLimit(String amount, String number) **Description:** Validates the accumulated amount of the given account number. **Parameter:** amount - the amount entered by user to be used for transaction number - the account number **Returns:** True if the accumulated amount value is valid

Class Transfer

---

class **Transfer**
extends **CommonFeatures**

This class contains a method call to askUser() in the CommonFeatures class prompting the user to input the account number to be transferred from, the account number to be transferred to and the account name. It makes sure to write every successful transaction into a transaction summary message in proper format.

Method Detail

Method and Description
private void transferAcct (String fAcct, String tAcct, String amount) **Description:** Stores the transaction messages of the form CC AAA BBB NNN MMM in the arraylist storing the transaction summary of the current banking session – CC is a two character transaction code in the form: *TR-transfer*

– AAA is the first (to) account number – BBB represents the (from) account number – NNN is the amount, in cents – MMM represents the field for the account name : *** **Parameters:** fAcct - the account number *amount* is being transferred to tAcct - the account number *amount* is transferred from amount - the amount to be transferred
public void transferStart() **Description:** Deals with the process of transferring money from one account to another.

Class Withdraw

---

class **Withdraw**
extends **CommonFeatures**

The withdraw class contains a method call to askUser() in the CommonFeatures class. It prompts the user to input the account number and amount to be withdrawn. It makes sure to write every successful transaction into a transaction summary message in proper format.

Method Detail

Method and Description
private void withdrawAcct (String number, String amount) **Description:** Stores the transaction messages of the form CC AAA BBB NNN MMM in the arraylist storing the transaction summary of the current banking session – CC is a two character transaction code in the form: *WD-withdrawal* – AAA is the first (to) account number in the form: *00000000* – BBB represents the (from) account number – NNN is the amount, in cents – MMM represents the field for the account name in the form *** **Parameters:** number - the account number *amount* is withdrawn from amount - the amount to be withdrawn
public void withdrawStart() **Description:** Deals with the money withdrawal process from a user's account.

Class Deposit

---

class **Deposit**
extends **CommonFeatures**

The deposit class contains a method call to askUser() in the CommonFeatures class. It prompts the user to input the account number and amount to be deposited. It makes sure to write every successful transaction into a transaction summary message in proper format.

Method Detail

Method and Description
private void depositToAcct(String number, String amt) **Description:** Stores the transaction messages of the form CC AAA BBB NNN MMM in the arraylist storing the transaction summary of the current banking session – CC is a two character transaction code in the form: *DE-deposit* – AAA is the first (to) account number – BBB represents the (from) account number in the form: *00000000* – NNN is the amount, in cents – MMM represents the field for the account name in the form: *** **Parameters:** number - the account number *amount* is deposited to amount - the amount to be deposited
public void depositStart() **Description:** This method deals with the process of depositing money to an account.

Class Logout

---

public class **Logout**
extends **CommonFeatures**

The Logout class contain a method call to simBank

A call to the Logout class signifies the end of the current banking session. It contains a call to simStart() method in the SimBank class. This ensures that the program does not stop running unless interrupted (possibly through the keyboard interrupt: *Ctrl* + *C*).

Method Detail

Method and Description
private void writeSummary() **Description:** This method writes to the transaction summary file whenever the user logs out out of a banking session
public void logoutStart() **Description:** This method deals with the process of logging out of a banking session