

Fancy ATM Design Documentation

Yufan Wen

U87145681

evanswen@bu.edu

Index

<u>FANCY ATM DESIGN DOCUMENTATION</u>	<u>1</u>
<u>INTRODUCTION</u>	<u>3</u>
<u>OBJECT ORIENTED DESIGN SPECIFICATION</u>	<u>3</u>
ACCOUNT	3
CHECKINGACCOUNT & SAVINGACCOUNT	3
CURRENCY	4
LOAN	4
TRANSACTION.....	5
DATE	6
GUIATM	6
GUILOGIN	6
GUIMANAGER	7
GUIQUERY	7
REPORT	7
ATMSHELL	7
<u>EXAMPLES</u>	<u>8</u>

Introduction

In this assignment, we are asked to implement a ATM terminal with design of Object Oriented Design and JAVA, in this assignment, I implemented function as transfer, deposit, create accounts, make loans and etc. Customer can check his/her own transaction records, and manager can view transaction records of all customers, and he can also check transaction filtered by date or customers.

Object Oriented Design Specification

Account

Benefit:

Abstract the account class, since different class might have different actions in the future, extract common attributes and methods both saving and checking accounts might have.

Meaning:

This abstract class extract common activities of an account, which contains different methods of an account:

Attributes:

String password;

String userName;

public Currency balance;

public boolean hasLoan;

public Loan loan;

public List<Transaction> transactions;

Methods:

Constructor():

public String getPassword(): get password of this account

public boolean setLoan(): check if this account has an loan from bank

public boolean payLoan(): method for account to pay his/her loan, if payment available, return true, else, return false

CheckingAccount & SavingAccount

Benefits:

Designed to perform different action that one account would have.

Meaning:

For different account type in this ATM, the activity is the same, so just put it in different classes, if there will be different activity for different classes, we can add them into these classes in order to expand function.

Currency

Benefits:

This ATM terminal provide function of exchanging different currency, and currency is basic unit of different action such as transaction, loan, so I designed a class named Currency to encapsulate these attribute and actions. Different class design is based on this class, since this class is an abstraction from the currency in real life.

Meaning:

This class defines different currency related activities including adding, delining, changing different currency types and change output to String type.

Attributes:

```
public Map<String,Double> rate = new HashMap<>();  
private String Cur_Cur;(means Current Currency)  
private Double lastCharge;  
private Double value;
```

Methods:

Constructor(): store different currency types and its rate in a map, so we can change Currency according to currency types.

public boolean setCurrency(): set different currency types for current account.

public boolean addValue(): add a number of money to current account

public boolean declineValue: decline a number of money from current account

public Currency mergeCur: while there are different currency types, call this method to merge them into main currency type the account is using.

public String toString()

public String getCur_Cur()

public Double getLastCharge()

public Double getValue()

Loan

Benefits:

Extract a loan class to specify attributes an loan might have, including its username, its amount, term, start date for presentation and calculate interest according to its term, thus making it much

clearer when interacting with currency of current account.

Meaning:

To store the loan record, and it will present in guiQuery, this class will help calculating the interests of a loan. Loan is calculated by the term, whose unit is month.

Attributes:

```
public String userName;  
public Currency currency;  
public Date startDate;  
public int termMonth;  
public Currency interest;  
public boolean isPaid;
```

Methods:

```
Constructor():  
public Double calculateInterest(): calculate interest the Bank might get from this loan.
```

Transaction

Benefits:

Class transaction extract basic attributes of an action of transaction, it is an encapsulation of basic attributes of transaction, and make it easier while generating the daily reports, since we can have every detail of one transaction and have it presented while calling Report class.

Meaning:

This class stores information of current transaction performed by current account.

Attributes:

```
public Date date;  
public String fromUser;  
public String toUser;  
public Currency value;  
public Currency charge;  
public Currency balanceLeft;  
public String description;
```

Methods:

```
Constructor()
```

Date

Benefits:

This class have less interaction to other classes than Currency, but it is still essential since the report need date information as prefix, so we have to call this class while we are calling reports, loan class.

Meaning:

This class generate today's date, for presentation in JTable.

Attributes:

```
private int day;  
private int month;  
private int year;
```

Methods:

```
public String toString(): change date parameters to Strings for the convenience of printing in  
table.  
public int getDay()  
public int getMonth()  
public int getYear()
```

guiATM

Benefits:

Called at the very beginning of action, decide next step of user.

Meaning:

Checking if account has login, if login, create a guiQuery, if not, then create a login user interface.

guiLogin

Benefits:

This gui class lead to different action of an user, if he/she already have an account, he can login or create another account, and if he/she is manager, he can also login to check the transaction informations.

Meaning:

This GUI class present user interface for login and create account, with the help of JTextField, JPasswordField, JButton.

guiManager

Benefits:

This class make an easy access to information a manager need.

Meaning:

This GUI class is for manager to examine transaction records of different users, he can directly enter this user interface because he has the highest authorization.

guiQuery

Benefits:

This class make an accessible user interface for current account, he/she can perform variety of action using this panel.

Meaning:

This is GUI class for current account to complete operation as transaction, deposit, withdraw, changing currency, and etc. whenever current user want to make some operation, it will open anther information window for user to input information required, and once user finished operation, it will provide feedback as information window.

Report

Benefits:

Using table to present detailed transaction of different users to manager, and personal transaction to current users just as bank in real life do.

Meaning:

This GUI class provide information of account's activities, user can only see his/her own different transactions, but manager can view all users' activities.

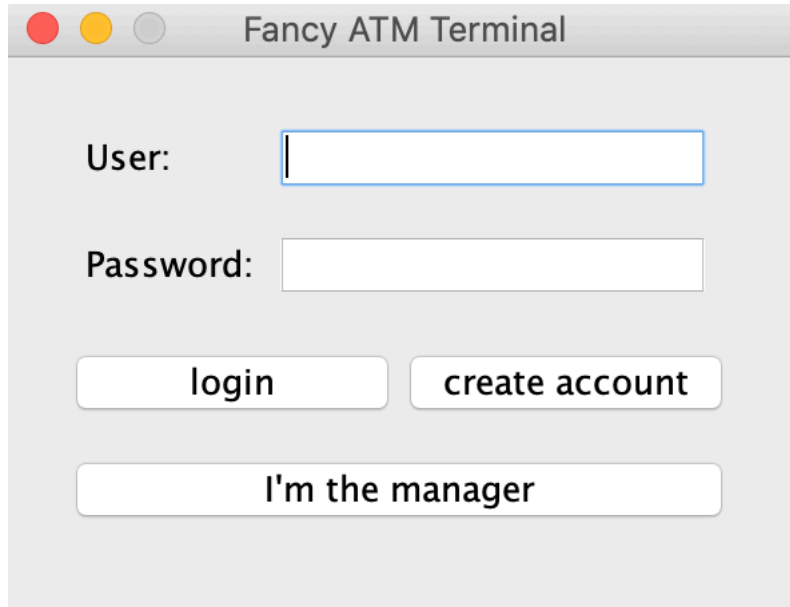
ATMShell

Meaning:

The main class that initialize the whole project.

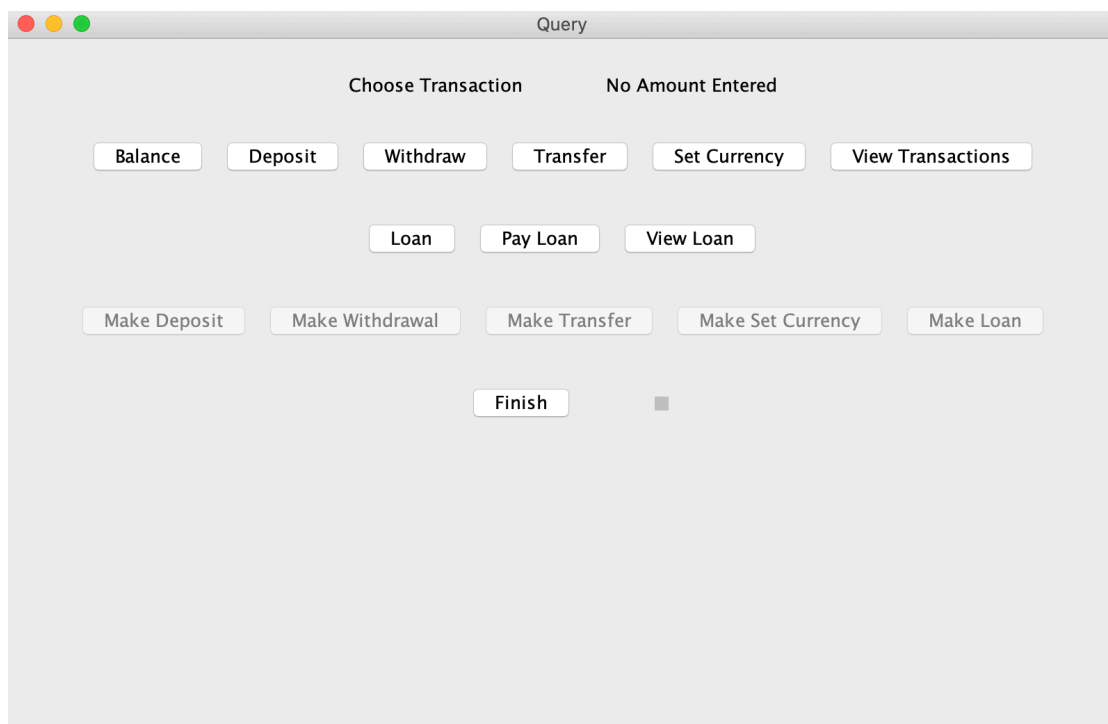
Examples

ATM terminal:



A screenshot of a window titled "Fancy ATM Terminal". The window has a light gray background and a title bar with three colored buttons (red, yellow, gray). It contains two input fields: "User:" with a blue border and "Password:" with a white border. Below the input fields are three buttons: "login", "create account", and "I'm the manager".

Query interface:



A screenshot of a window titled "Query". The window has a light gray background and a title bar with three colored buttons (red, yellow, green). It contains a grid of buttons organized into two columns: "Choose Transaction" and "No Amount Entered". The "Choose Transaction" column includes buttons for "Balance", "Deposit", "Withdraw", "Transfer", "Loan", "Pay Loan", "View Loan", "Make Deposit", "Make Withdrawal", "Make Transfer", "Make Set Currency", and "Make Loan". The "No Amount Entered" column includes buttons for "Set Currency", "View Transactions", and "Finish". A small gray square is located below the "Finish" button.

Personal report

