CSE201 Term Project (Fall 2023) (Version 2)

-Automated Teller Machine (ATM) System-

Submission Deadline: 2023/12/1 (FRI)

In-Person Live Demo: 2023/12/4(MON) or 2023/12/6(WED) class time

General Instruction

This is a term project that a group of students work together. Every member in a group must show full dedication to the project by actively collaborating with other members. The total grading points are **100 pt** for each member student. Note that each member in a group may be graded differently, if there are unequal contributions among team members. Copying, cheating or any other academic misconduct is strictly prohibited.

In this term project, you will be given an imaginary system description that your group should implement **using the concept of the object-oriented programming paradigm**. In addition, the system requirements are also given to provide what functionalities shall be implemented in general. Since the purpose of the system requirements is to provide you with general guidance of the system implementation, it may not include all details needed to implement the system. For this reason, you may need to reasonably use your own judgement to fill the gap between the requirements and your implementation.

In addition, we do not provide any specific test scenarios to validate the correctness of your implementation. **It is your job to create adequate test scenarios based on the system requirements** in order to claim the correctness of your implementation.

Group Forming Instruction

- Students are encouraged to form a group of size 2 – 4 to start the term project.
- Even though it is advised to form a group to learn a collaborative work, it is also ok to work independently. There is no additional credit or penalty for the independent project.
- If some students want to form a group, but have difficulty in finding group members, contact the instructor. The instructor will gather information from those students, and try to help forming a group.

Overall System Descriptions

An automated teller machine (ATM) is computerized telecommunications device that provides a financial institution's (bank) customers a method of performing financial transactions, in a public space without the need for a human bank teller. Through ATM, customers interact with a user-friendly interface that enables them to access their bank accounts and perform various transactions.

Fig 1 illustrates the scope of the ATM system. In order to use an ATM, a user (customer) must have a bank account first, and a bank issues a (debit) card that can be used to access the bank account through ATMs. Note that ATMs do not maintain any user information locally such as account numbers or passwords. For this reason, ATMs are connected to external banks to retrieve and verify necessary user information to perform transaction requests.

A user may access several different types of ATMs. Each ATM is managed by a primary bank, and some ATMs may allow users to perform transactions with other non-primary bank accounts as well (e.g., withdraw funds from Daegu Bank account using KookMin Bank ATM). An ATM keeps a certain amount of available cash that can be immediately provided to users. The available cash inside ATMs may increase or decrease as multiple users perform a series of transactions such as deposit or withdraw cash. An ATM also allows non-cash transactions such as bank transfers or check deposits.

In this project, you will be given the system requirements of the three subsystems, ATM, Bank and Account, and your goal is to implement them so that a user can perform bank transactions via ATMs, especially using the concept of the object-oriented programming paradigm.

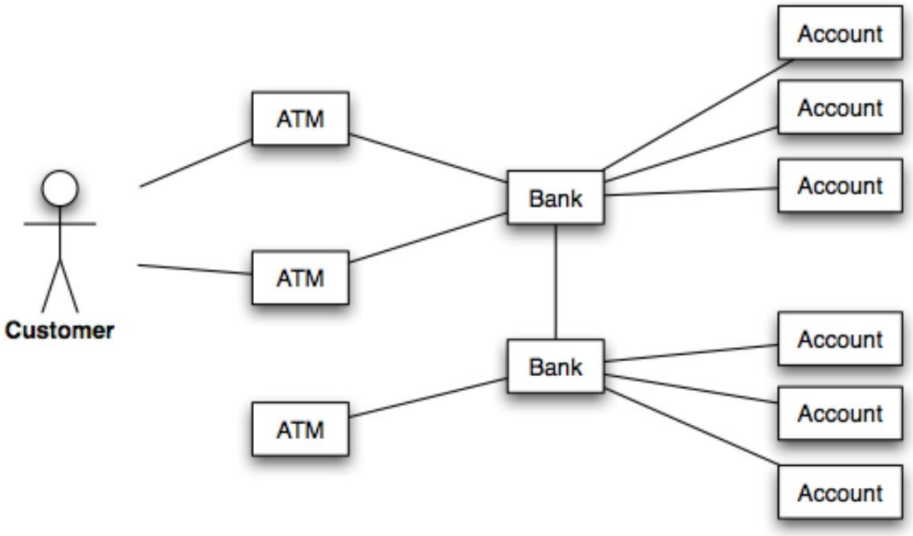

Figure 1 UML Use Case Diagram

Definitions

Term	Definition
User	A person who is using an ATM machine
ATM	Automated Teller Machine
Bank	A bank system that holds all account information
Account	An account that can hold a user's fund information
Account Number	A unique numeric number associated with an account
Password	A password associated with the Card and Account
Card	An access that can be used to perform transactions with ATMs
Cash	A paper cash
Check	A paper check that is printed with its amount
Card Insert Slot	A slot where a card can be inserted
Cash/Check Insert and Dispenser Slot	A slot where a cash or check can be inserted. The same slot can be also used to dispense cash
Transaction	A bank service that can be performed with ATM such as deposit, withdrawal, transfer.
Sessions	A logical period within which transactions can be performed
Primary Bank	A bank that manages a certain ATM
Non-primary Bank	A bank that does not manage a certain ATM
Fee	A cost that is applied for each transaction
Serial Number	A unique numeric number associated with the ATM
Single-Bank ATM	An ATM that can be used by one bank brand (e.g., Daegu Bank only)
Multi-Bank ATM	An ATM that can be used by multiple bank brands (e.g., KookMin and Daegu Banks)
Admin Card	A special type of card with which a bank administrator can access an ATM

User Interface Requirement

Typical ATM has a user interface with which a customer can perform various banking services such as deposit, withdraw or transfer. Figure 2 shows one example user interface of ATMs.

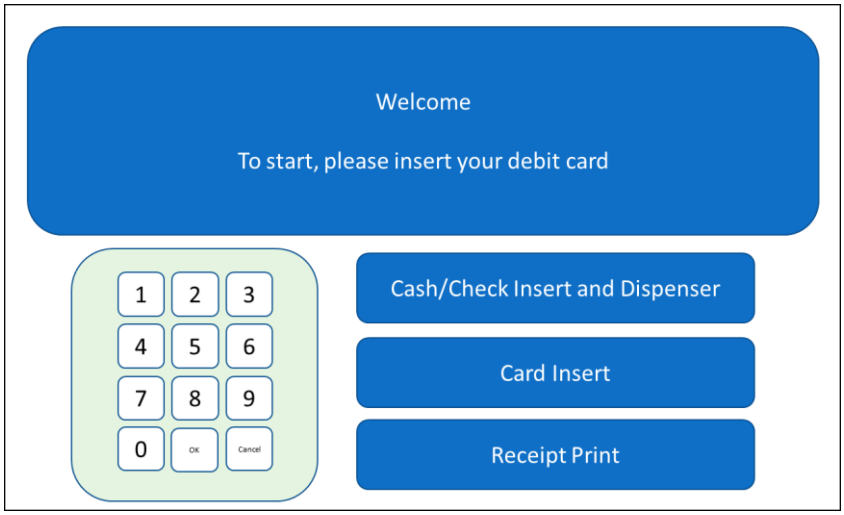

Figure 2 Example of ATM User Interface

More specifically, a customer can give input or receive output from the ATMs as follows:

- **Display panel**: Necessary information or direction for each banking service is displayed in text. Some ATMs are bilingual for international uses such as English and Korean.
- **Keypad**: 10 digits (0 – 9) are used to give numeric input such as the amount of money, passcode, bank account numbers and so on. In addition, "OK" and "Cancel" buttons are used to proceed or revert the customer actions in the process of banking services.
- **Cash/Check Insert and Dispenser Slot**: Cash can be inserted or dispensed via this slot. The slot may take different types of cash available in South Korea including KRW 1,000, 5,000, 10,000, 50,000. In addition, the slot can take bank checks as well. The minimum amount of the bank check is KRW 100,000; that is, any amount above KRW 100,000 is considered a valid bank check (e.g., KRW 100,000, 100,001, 234,567 are all valid checks). The limit of the number of cash or checks that can be inserted or dispensed at a time is 50.
- **Card Insert Slot**: A customer can insert her or his debit card. ATM should have an appropriate function to verify the inserted card (e.g., checking a passcode).
- **Receipt Print Slot**: A customer may request a receipt that summarize bank service transactions, and the receipt is given via this slot.

In this project, the input/output user interface is implemented **using the console input/output**. All functions that can be done via the above user interface should be appropriately implemented via the console input/output. For example, inserting a card to the slot can be implemented by entering the card number in the console input.

System Features

1. System Setup

- Description
  - The system consists of ATMs and Banks and Accounts, and the three parts shall be appropriately initialized before being used by users.
- Functional Requirement
  - (REQ1.1) An ATM has a 6-digit serial number that can be uniquely identified among all ATMs (e.g., 315785).
  - (REQ1.2) An ATM is set to one of the following types: (1) Single Bank ATM, (2) Multi-Bank ATM.
    - For Single Bank ATM, the ATM is belonged to a primary bank, and only a card issued by the primary bank is considered valid.
    - For Multi-Bank ATM, there is a primary bank that manages the ATM, but a card issued by any other banks is considered valid.
  - (REQ1.3) An ATM may support either unilingual or bilingual languages.
    - When an ATM is configured unilingual, all information is displayed in English only.
    - When an ATM is configured bilingual, a user can choose if the information is to be displayed either English or Korean (**Note**: if you know only one of the languages, consider using a language translation service, such as Google Translation).
  - (REQ1.4) A Bank deposits a certain amount of cashes to an ATM to serve users.
  - (REQ1.5) A Bank can open an Account for user with necessary information to perform bank services.
    - (e.g.) Bank name (e.g, Kakao, Shinhan), User name, Account number (12-digit), Available funds, Transaction histories.
  - (REQ1.6) A user may have multiple Accounts in a Bank.
  - (REQ1.7) A user may have Accounts in multiple Banks.
  - (REQ1.8) Each ATM have several types of transaction fees, and paid as follows:
    - Deposit fee for non-primary banks: KRW 1,000; the fee is paid by inserting additional cash.
    - Deposit fee for primary banks: KRW 0
    - Withdrawal fee for a primary bank: KRW 1,000; the fee is paid from the withdrawal account.
    - Withdrawal fee for non-primary banks: KRW 2,000; the fee is paid from the withdrawal account.
    - Account transfer fee between primary banks: KRW 2,000; the fee is paid from the source account.
    - Account transfer fee between primary and non-primary banks: KRW 3,000; the fee is paid from the source account.
    - Account transfer fee between non-primary banks: KRW 4,000; the fee is paid from the source account.
    - Cash transfer fee to any bank type: KRW 5,000; the fee is paid by inserting additional cash.

- (REQ1.9) An admin can access the menu of "Transaction History" via an admin card (See REQ Display of Transaction History).
- (REQ1.10) An ATM only accepts and returns the following types of cashes and checks.
  - (Cash type) KRW 1,000, KRW 5,000, KRW 10,000, KRW 50,000
  - (Check type) Any amount over KRW 100,000 check (e.g., KRW 100,000, 100,001, 234,567 are all valid checks)
- (REQ1.11) All accounts and ATMs shall be created and initialized during the program execution.
  - During the program execution, the necessary information to create accounts and ATMs shall be given from a user via console input (*i.e.*, hard coding of account and ATM information is not allowed).
  - The accounts and ATMs shall be created and initialized based on the user input.

2. ATM Session

- Description
  - An ATM maintains a session during which a user can make multiple transactions.
- Functional Requirement
  - (REQ2.1) A session starts when a user inserts a card.
  - (REQ2.2) A session ends whenever a user wishes (e.g., by choosing a cancel button) or there are some exceptional conditions detected by the ATM (e.g., no cash available).
  - (REQ2.3) When a session ends, the summary of all transactions performed in a session must be displayed.
    - (e.g.) Account/card info, transaction types (deposit, transfer, withdrawal), and their amount, …
  - (REQ2.4) Each transaction has a unique identifier across all sessions.

3. User Authorization

- Description
  - An ATM shall authorize a user before initiating any transactions. This authorization occurs after a user starts a new session by inserting a valid card, and before any transaction is started.
  - An ATM does not have the user information locally, so it needs to communicate with a Bank to authorize the user.
  - Once a user is authorized, a user may perform multiple transactions without any further authorization until the session ends.
- Functional Requirement
  - (REQ3.1) An ATM checks if the inserted card is valid for the current type of ATM.
    - See REQ in System Setup which card is considered valid
  - (REQ3.2) If an invalid card is inserted, the ATM shall display an appropriate error message (e.g., Invalid Card).
  - (REQ3.3) An ATM shall ask a user to enter the password (e.g., Enter Password), and verify if the password is correct.

- An ATM does not maintain any user information, so the card and password information need to be passed to the bank; then, the bank verifies the password, and return the authorization result.

- (REQ3.4) If the entered password is incorrect, the ATM shall display an appropriate error message (e.g., Wrong Password).
- (REQ3.5) If a user enters wrong passwords 3 times in a row, a session is aborted, and return the card to the user.

4. Deposit

- Description
  - A user can deposit cash or checks to an ATM via Cash/Check Insert and Dispenser Slot.
  - There is a limit in the number of cash or checks that can be deposited per transaction (e.g., 50 paper cashes, 30 paper checks)
- Functional Requirement
  - (REQ4.1) An ATM shall take either cash or check from a user.
  - (REQ4.2) An ATM shall display an appropriate error message if the number of the inserted cash or checks exceed the limit allowed by the ATM.
  - (REQ4.3) Once cash or checks are accepted by ATM, the transaction must be reflected to the bank account as well (i.e., the same amount of fund must be added to the corresponding bank account).
  - (REQ4.4) Some deposit fee may be charged (See REQ in System Setup)
  - (REQ4.5) The deposited cash increase available cash in ATM that can be used by other users.
  - (REQ4.6) The deposited check does not increase available cash in ATM that can be used by other users.

5. Withdrawal

- Description
  - A user can withdraw cash only from an ATM by specifying the amount. A check cannot be used for withdrawal.
  - There is a limit in the number of withdrawals of fund per session.
  - There is a limit in the amount of withdrawal of fund per transaction.
- Functional Requirement
  - (REQ5.1) An ATM shall ask a user to enter the amount of fund to withdraw.
  - (REQ5.2) An ATM shall display an appropriate error message if there is insufficient fund in the account or insufficient cash in the ATM.
  - (REQ5.3) Once the withdrawal is successful, the transaction must be reflected to the bank account as well (i.e., the same amount of fund must be deducted from the corresponding bank account).
  - (REQ5.4) Some withdrawal fee may be charged (See REQ in System Setup).
  - (REQ5.5) The cash withdrawal lower available cash in the ATM that can be used by other users.
  - (REQ5.6) The maximum number of withdrawals per each session is 3.

- If a user wants to withdraw four times, it needs to end the current session after withdrawing three times and restart another session for one more withdrawal.

- (REQ5.7) The maximum amount of cash withdrawal per transaction is KRW 500,000.

6. Transfer

- Description
  - (Cash transfer) A user can transfer cash to another account (cash needs to be inserted to ATM for this transfer).
  - (Account transfer) A user can transfer available fund in one account to another account (no cash insertion to ATM is needed for this transfer).
- Functional Requirement
  - (REQ6.1) An ATM shall ask a user to choose the transfer types either cash transfer or account fund transfer.
  - (REQ6.2) For both cash and account transfers, an ATM shall ask the destination account number where the fund is to be transferred.
  - (REQ6.3) For cash transfer, an ATM shall ask user to insert the cash including the transaction fees, and verify if the amount of the inserted cash is correct. All inserted cash excluding the transaction fee shall be transferred.
  - (REQ6.4) For account transfer, an ATM shall ask the source account number, and the amount of fund to be transferred.
  - (REQ6.5) Some transfer fee may be charged (See REQ in System Setup).
  - (REQ6.6) The inserted cash for transfer increase available cash in ATM that can be used by other users.
  - (REQ6.7) Once the transfer is successful, the transaction must be reflected to the bank account as well (i.e., the same amount of fund must be deducted from the source bank account, and then added to the destination bank account).

7. Display of Transaction History (Admin Menu)

- Description
  - An administrator of an ATM may request an ATM to display the whole transaction histories of all users from the beginning of the system start.
- Functional Requirement
  - (REQ7.1) When a session is started by an admin by inserting an admin card (See REQ in System Setup), an ATM displays a menu of “Transaction History” only.
  - (REQ7.2) When the “Transaction History” menu is selected, an ATM display the information of all transactions from all users from the beginning of the system start.
    - Transaction ID, Card Number, Transaction Types, Amount, other transaction-specific information
    - Each transaction may have different types of information, so they need to be appropriately displayed (e.g., a deposit transaction does not have the source account information in a transfer transaction).
  - (REQ7.3) The “Transaction History” information shall be outputted to the external file (e.g., txt

file).

8. Multi-language support

- Description
  - An ATM may be configured either unilingual or bilingual (See REQ in System Setup).
  - When it is configured bilingual, the ATM gives an option to choose the language to display either English or Korean.
  - All menus are displayed with the chosen languages.
- Functional Requirement
  - (REQ8.1) An ATM that is configured with the bilingual support shall provide an option for a user to choose the preferred language either English or Korean.
  - (REQ8.2) Once a certain language is chosen, all menus must be displayed using the chosen language.

9. Exception Handling

- Description
  - A user may encounter an exceptional scenario while using an ATM such as low cash in ATM, incorrect password, invalid cash types, exceed the maximum number of withdrawals, low ATM cash, and so on.
  - Some exception scenarios are described in the respective requirements, but there are other exception scenarios that are not clearly stated in the document such as an invalid input range.
- Functional Requirement
  - (REQ9.1) An ATM shall display an appropriate message for each exception scenario (both explicitly stated in this document and implicitly assumed ones), and take an appropriate action (e.g., print an appropriate error message, end a session).

10. Display of Account/ATM Snapshot

- Description
  - This requirement is to facilitate the term project grading process.
  - At any point of the program execution, the program shall be able to print information of all existing accounts and ATMs.
- Functional Requirement
  - (REQ10.1) When a particular character (e.g. '**x**') is given as a console input during the program execution, the following information shall be displayed to the console.
    - All ATMs' information: Remaining cash
      - (e.g., ATM [SN: 111111] remaining cash: 7000, ATM [SN: 222222] remaining cash: 4000, ATM [SN: 333333] remaining cash: 2000)
    - All accounts' information: Remaining balance
      - (e.g., Account [Bank: Kakao, No: 111111111111, Owner: Jenny] balance: 7000, Account [Bank: Daegu, No: 222222222222, Owner: Tom] balance: 1000, Account [Bank: Shinhan, No: 333333333333, Owner: Jenny] balance: 2000)

Submission Deadline and Deliverables

- **Deliverables**
  - **Deadline: 2023/12/1 (FRI)**
  - Create a single zip file that includes the following files.
  - **Submission 1**: the final report (pdf, word, hwp)
    1. List the names and student IDs for all team members
    2. The final class diagram design.
    3. The list of requirements that have been successfully implemented; the list must specify the requirement ID (e.g., REQ1.2).
    4. The console screenshots with explanations to demonstrate the successfully implemented requirements: each screenshots must specify the requirement ID (e.g., REQ1.2), and each screenshot must accompany with a brief explanation why the selected test cases are reasonable to test each requirement; there will be some penalty, if it does not come with a brief explanation.
    5. The list of concepts of object-oriented programming (e.g., inheritance, polymorphism), and explanation how they are used.
    6. Instruction to run the source code
    7. The final version of source code (copy and paste from cpp file)
    8. Member student contribution table and note (e.g., Student A: 50%, Student B: 20% or all students equally contributed)
  - **Submission 2**
    1. The final source code (cpp files)
- **In-Person Live Demo**
  - **2023/12/4(MON) or 2023/12/6(WED) during the class time**
  - There will be an in-person live demo to check the correctness of the program.
    1. At least one team member should bring a laptop to the in-person demo location (the location will be announced later).
    2. The instructor or TA will give a set of test cases, and the team member should run the program accordingly. The correctness of the program will be checked by comparing the obtained output to the expected output.
    3. If the program crashes during the demo or does not have the feature to execute the test case, the base score will be assigned.

(Tentative) General Grading Criteria

(NOTE) This is the example of the grading criteria to guide the term project at the early stage. The final criteria may be different from these examples. Some criteria may be added or removed to/from the current version.

- Correctness of the Program
  - How well the test scenarios are designed to show the requirement conformance?
  - Does the program work according to the requirement?
- Applying Object-Oriented Programming Concept
  - How well you used the encapsulation concept?
  - How well you abstracted the system?
  - How well you used the inheritance concept?
  - How well you the polymorphism concept?
  - How well you used the exception handling?
  - How well you used STL?
- Presentation
  - The quality of the presentation
  - The quality of the final report