Code for the ATM System

Write the object-oriented code to implement the design of the ATM problem.

We'll cover the following

We've covered different aspects of the ATM and observed the attributes attached to the problem using various UML diagrams. Let's now explore the more practical side of things where we will work on implementing the ATM using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in the ATM system:

- Java
- C#
- Python
- C++
- JavaScript

ATM classes

In this section, we will provide the skeleton code of the classes designed in the class diagram lesson.

> **Note:** For simplicity, we are not defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public method functions.

Enumerations

The following code provides the definition of the enumeration used in the ATM system.

`ATMStatus`: This enumeration keeps track of the following states of an ATM:

- Idle
- Card inserted by the user
- Option selected
- Cash withdrawal
- Money transfer
- Display the account balance

> **Note:** JavaScript does not support enumerations, so we will be using the `Object.freeze()` method as an alternative that freezes an object and prevents further modifications.

```
// Enumeration
enum ATMStatus {
    Idle,
    HasCard,
    SelectionOption,
    Withdraw,
    TransferMoney,
    BalanceInquiry
}
```

Definition of the ATMState enum

User and ATM card

The `User` class stores the user's `ATMCard` and bank account, where the `ATMCard` class holds the card number, customer name, card expiration date, and PIN. The definitions of these classes are provided below:

```
public class User {
    private ATMCard card;
    private BankAccount account;
}

public class ATMCard {
    private String cardNumber;
    private String customerName;
    private Date cardExpiryDate;
    private int pin;
}
```

The User and ATMCard classes

Bank and bank account

The `Bank` class represents a bank having a name and code and can also add an ATM. The `BankAccount` class represents a bank account that has two child classes: `SavingAccount` and `CurrentAccount`. These derived classes have a method for finding the withdrawal limit. The definitions of these classes are provided below:

```
public class Bank {
    private String name;
    private String bankCode;

    public String getBankCode();
    public boolean addATM();
}

public class BankAccount {
    private int accountNumber;
    private double totalBalance;
    private double availableBalance;

    public double getAvailableBalance();
}

public class SavingAccount extends BankAccount {
    public double withdrawLimit();
}

public class CurrentAccount extends BankAccount {
    public double withdrawLimit();
}
```

The Bank and BankAccount classes

Card reader, card dispenser, printer, screen, and keypad

The `CardReader`, `CashDispenser`, `Keypad`, `Screen` and `Printer` classes compose the ATM and have the following functionalities:

- `CardReader`: It reads the card inserted by the user.
- `CashDispenser`: It dispenses cash upon withdrawal request.
- `Keypad`: It is used by the user to enter the PIN for authentication.
- `Screen`: It displays messages.
- `Printer`: It prints receipts.

The definitions of these classes are provided below:

```
public class CardReader {
    public boolean readCard();
}

public class CashDispenser {
    public boolean dispenseCash();
}

public class Keypad {
    public String getInput();
}

public class Screen {
    public void showMessage();
}

public class Printer {
    public void printReceipt();
}
```

The CardReader, CashDispenser,Keypad, Screen, Printer classes

ATM state

`ATMState` is an abstract class that is extended by `IdleState`, `HasCardState`, `SelectOperationState`, `CheckBalanceState`, `CashWithdrawalState` and `TransferMoneyState`. All of these derived classes override the `returnCard()` and `exit()` functions of the `ATMState` class. The derived classes individually override the following functions:

- `IdleState`: This class overrides the `insertCard()` function.
- `HasCardState`: This class overrides the `authenticatePin()` function.
- `SelectOperationState`: This class overrides the `selectOperation()` function.
- `CheckBalanceState`: This class overrides the `displayBalance()` function.
- `CashWithdrawalState`: This class overrides the `cashWithdrawal()` function.
- `TransferMoneyState`: This class overrides the `transferMoney()` function.

The definitions of these classes are provided below:

```
public abstract class ATMState {

    public abstract void insertCard(ATM atm, ATMCard card);

    public abstract void authenticatePin(ATM atm, ATMCard card, int pin);

    public abstract void selectOperation(ATM atm, ATMCard card, TransactionType tType);

    public abstract void cashWithdrawal(ATM atm, ATMCard card, int withdrawAmount);

    public abstract void displayBalance(ATM atm, ATMCard card);

    public abstract void transferMoney(ATM atm, ATMCard card, int accountNumber, int transferAmount);

    public abstract void returnCard();

    public abstract void exit(ATM atm);

}

public class IdleState extends ATMState {

    @Override
    public void insertCard(ATM atm, ATMCard card) {
        // definition
    }

    @Override
    public void authenticatePin(ATM atm, ATMCard card, int pin) {
        // definition
```

The ATMState and its derived classes

ATM and ATM room

An `ATMRoom` has an `ATM` and a `User` with the following:

- A specific state at a given moment
- Balance
- A limited number of hundred, fifty, and ten dollar bills

The definitions of these classes are provided below:

```
public class ATM {
    private static ATM atmObject = new ATM(); //Singleton
    private ATMState currentATMState;
    private int atmBalance;
    private int noOfHundredDollarBills;
    private int noOfFiftyDollarBills;
    private int noOfTenDollarBills;

    // References to various ATM components
    private CardReader cardReader;
    private CashDispenser cashDispenser;
    private Keypad keypad;
    private Screen screen;
    private Printer printer;

    public void displayCurrentState();
    public void initializeATM(int atmBalance, int noOfHundredDollarBills, int noOfFiftyDollarBills, int noOfTenDollarBills) {
}

public class ATMRoom {
    private ATM atm;
    private User user;
}
```

The ATM and ATMRoom classes

Wrapping up

We've explored the complete design of the ATM in this chapter. We've looked at how a basic ATM system can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.

Back

Activity Diagram for the ATM System

Complete

Next

Getting Ready: The Chess Game