Back To Course Home

Grokking the Object Oriented Design Interview

0% completed

Search Course

Object-Oriented Design and UML
- Object-Oriented Basics
- OO Analysis and Design
- What is UML?
- Use Case Diagrams
- Class Diagram
- Sequence diagram
- Activity Diagrams

Object Oriented Design Case Studies
- Design a Library Management System
- Design a Parking Lot
- Design Amazon - Online Shopping System
- Design Stack Overflow
- Design a Movie Ticket Booking System
- Design an ATM
- Design an Airline Management System
- Design Blackjack and a Deck of Cards
- Design a Hotel Management System
- Design a Restaurant Management system
- Design Chess
- Design an Online Stock Brokerage System
- Design a Car Rental System
- Design LinkedIn
- Design Cricinfo
- Design Facebook - a social network

Appendix
- Contact us
- Other courses

Course Certificate

Mark Course as Completed

Design an ATM

We'll cover the following

- Requirements and Goals of the System
- How ATM works?
- Use cases
- Class diagram
- Activity Diagram
- Sequence Diagram
- Code

An automated teller machine (ATM) is an electronic telecommunications instrument that provides the clients of a financial institution with access to financial transactions in a public space without the need for a cashier or bank teller. ATMs are necessary as not all the bank branches are open every day of the week, and some customers may not be in a position to visit a bank each time they want to withdraw or deposit money.

ATM

Requirements and Goals of the System

The main components of the ATM that will affect interactions between the ATM and its users are:

1. **Card reader:** to read the users' ATM cards.
2. **Keypad:** to enter information into the ATM e.g. PIN. cards.
3. **Screen:** to display messages to the users.
4. **Cash dispenser:** for dispensing cash.
5. **Deposit slot:** For users to deposit cash or checks.
6. **Printer:** for printing receipts.
7. **Communication/Network Infrastructure:** it is assumed that the ATM has a communication infrastructure to communicate with the bank upon any transaction or activity.

The user can have two types of accounts: 1) Checking, and 2) Savings, and should be able to perform the following five transactions on the ATM:

1. **Balance inquiry:** To see the amount of funds in each account.
2. **Deposit cash:** To deposit cash.
3. **Deposit check:** To deposit checks.
4. **Withdraw cash** To withdraw money from their checking account.
5. **Transfer funds:** To transfer funds to another account.

How ATM works?

The ATM will be managed by an operator, who operates the ATM and refills it with cash and receipts. The ATM will serve one customer at a time and should not shut down while serving. To begin a transaction in the ATM, the user should insert their ATM card, which will contain their account information. Then, the user should enter their Personal Identification Number (PIN) for authentication. The ATM will send the user's information to the bank for authentication; without authentication, the user cannot perform any transaction/service.

The user's ATM card will be kept in the ATM until the user ends a session. For example, the user can end a session at any time by pressing the cancel button, and the ATM Card will be ejected. The ATM will maintain an internal log of transactions that contains information about hardware failures; this log will be used by the ATM operator to resolve any issues.

1. Identify the system user through their PIN.
2. In the case of depositing checks, the amount of the check will not be added instantly to the user account; it is subject to manual verification and bank approval.
3. It is assumed that the bank manager will have access to the ATM's system information stored in the bank database.
4. It is assumed that user deposits will not be added to their account immediately because it will be subject to verification by the bank.
5. It is assumed the ATM card is the main player when it comes to security; users will authenticate themselves with their debit card and security pin.

Use cases

Here are the actors of the ATM system and their use cases:

Operator: The operator will be responsible for the following operations:

1. Turning the ATM ON/OFF using the designated Key-Switch.
2. Refilling the ATM with cash.
3. Refilling the ATM's printer with receipts.
4. Refilling the ATM's printer with INK.
5. Take out deposited cash and checks.

Customer: The ATM customer can perform the following operations:

1. Balance inquiry: the user can view his/her account balance.
2. Cash withdrawal: the user can withdraw a certain amount of cash.
3. Deposit funds: the user can deposit cash or checks.
4. Transfer funds: the user can transfer funds to other accounts.

Bank Manager: The Bank Manager can perform the following operations:

1. Generate a report to check total deposits.
2. Generate a report to check total withdrawals.
3. Print total deposits/withdrawal reports.
4. Checks the remaining cash in the ATM.

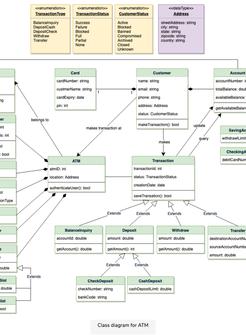
Here is the use case diagram of our ATM system:

ATM use case diagram

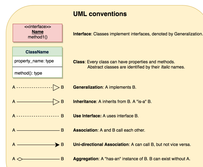
Class diagram

Here are the main classes of the ATM System:

- **ATM:** The main part of the system for which this software has been designed. It has attributes like 'atmID' to distinguish it from other available ATMs, and 'location' which defines the physical address of the ATM.
- **CardReader:** To encapsulate the ATM's card reader used for user authentication.
- **CashDispenser:** To encapsulate the ATM component which will dispense cash.
- **Keypad:** The user will use the ATM's keypad to enter their PIN or amounts.
- **Screen:** Users will be shown all messages on the screen and they will select different transactions by touching the screen.
- **Printer:** To print receipts.
- **DepositSlot:** User can deposit checks or cash through the deposit slot.
- **Bank:** To encapsulate the bank which owns the ATM. The bank will hold all the account information and the ATM will communicate with the bank to perform customer transactions.
- **Account:** We'll have two types of accounts in the system: 1)Checking and 2)Saving.
- **Customer:** This class will encapsulate the ATM's customer. It will have the customer's basic information like name, email, etc.
- **Card:** Encapsulating the ATM card that the customer will use to authenticate themselves. Each customer can have one card.
- **Transaction:** Encapsulating all transactions that the customer can perform on the ATM, like BalanceInquiry, Deposit, Withdraw, etc.

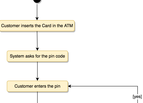


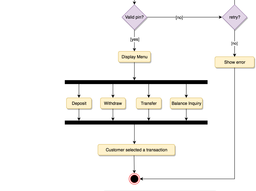
Class diagram for ATM



Activity Diagram

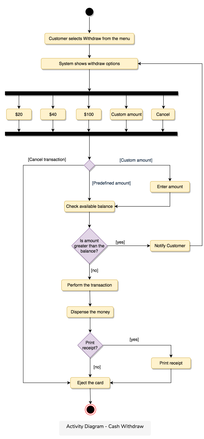
Customer authentication: Following is the activity diagram for a customer authenticating themselves to perform an ATM transaction:





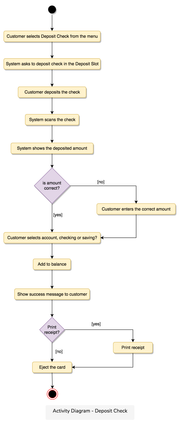
Activity Diagram - Customer Authentication

Withdraw: Following is the activity diagram for a user withdrawing cash:



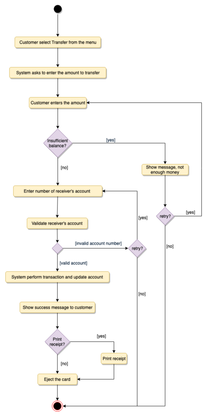
Activity Diagram - Cash Withdraw

Deposit check: Following is the activity diagram for the customer depositing a check:



Activity Diagram - Deposit Check

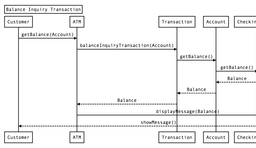
Transfer: Following is the activity diagram for a user transferring funds to another account:



Activity Diagram - Transfer funds

Sequence Diagram

Here is the sequence diagram for balance inquiry transaction:



Code

Here is the skeleton code for the classes defined above:

Enums and Constants: Here are the required enums, data types, and constants:

Java | Python

```java
public enum TransactionType {
  BALANCE_INQUIRY, DEPOSIT_CASH, DEPOSIT_CHECK, WITHDRAW, TRANSFER
}

public enum TransactionStatus {
  SUCCESS, FAILURE, BLOCKED, FULL, PARTIAL, NONE
}

public enum CustomerStatus {
  ACTIVE, BLOCKED, BANNED, COMPROMISED, ARCHIVED, CLOSED, UNKNOWN
}

public class Address {
  private String streetAddress;
  private String city;
  private String state;
  private String zipCode;
  private String country;
}

```

Customer, Card, and Account: "Customer" encapsulates the ATM user, "Card" the ATM card, and "Account" can be of two types: checking and savings:

Java | Python

```java
// 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 method and modified only through their public setter function.

public class Customer {
  private String name;
  private String email;
  private String phone;
  private Address address;
  private CustomerStatus status;

  private Card card;
  private Account account;

  public boolean makeTransaction(Transaction transaction);
  public Address getBillingAddress();
}

public class Card {
  private String cardNumber;
  private String customerName;
  private Date cardExpiry;
  private int pin;

  public Address getBillingAddress();
}

public class Account {
```

Bank, ATM, CashDispenser, Keypad, Screen, Printer and DepositSlot: The ATM will have different components like keypad, screen, etc.

Java | Python

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

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

public class ATM {
  private int atmID;
  private Address location;

  private CashDispenser cashDispenser;
  private Keypad keypad;
  private Screen screen;
  private Printer printer;
  private CheckDeposit checkDeposit;
  private CashDeposit cashDeposit;

  public boolean authenticateUser();
  public boolean makeTransaction(Customer customer, Transaction transaction);
}

public class CashDispenser {
  private int totalFiveDollarBills;
```

```
  private int totalTwentyDollarBills;

  public boolean dispenseCash(double amount);
```

Transaction and its subclasses: Customers can perform different transactions on the ATM, these classes encapsulate them:

Java | Python

```
public abstract class Transaction {
  private int transactionId;
  private Date creationTime;
  private TransactionStatus status;
  public boolean makeTransaction();
}

public class BalanceInquiry extends Transaction {
  private int accountId;
  public double getAccountId();
}

public abstract class Deposit extends Transaction {
  private double amount;
  public double getAmount();
}

public class CheckDeposit extends Deposit {
  private String checkNumber;
  private String bankCode;

  public String getCheckNumber();
}

public class CashDeposit extends Deposit {
  private double cashDepositLimit;
}
```

Back
Design a Movie Ticket Booking System

Next
Design an Airline Management System

Mark as Completed

Report an Issue
Ask a Question