.

# Getting Ready: The ATM System

Understand the ATM design problem and learn the questions to simplify this problem further.

**We'll cover the following**

* [Problem definition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Problem-definition)
* [Expectations from the interviewee](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Expectations-from-the-interviewee)
  + [ATM components](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ATM-components)
  + [ATM features](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ATM-features)
  + [ATM processing](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ATM-processing)
* [Design approach](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-approach)
* [Design patterns](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-patterns)

## Problem definition

An **Automated teller machine (ATM)** allows a bank customer to perform financial transactions remotely without the need for a teller or a bank branch. These transactions include deposits, withdrawals, balance inquiries, and account transfers. ATMs are located in convenient locations such as banks, grocery stores, airports, and other public places.

To use an ATM, users need a bank card associated with an account at a financial institution to use an ATM. To access their account, users must also know the personal identification number (PIN). The user can follow the prompts on the ATM's screen to complete the desired transaction with the PIN.

The user inserts their card in the ATM

**1** of 18

## Expectations from the interviewee

There are several components in an ATM design, each with specific constraints and requirements. The following provides an overview of some of the main expectations that the interviewer will want to hear you discuss in more detail, during the interview.

### ATM components

To better understand an ATM system, you may ask the interviewer the following questions:

1. What are the components of an ATM?
2. Is the ATM necessarily placed inside a room?
3. Does an ATM have a fingerprint scanner?

### ATM features

Different ATMs may vary in terms of features which is why it is important to clear the following questions from the interviewer:

1. What is the withdrawal limit of an ATM?
2. Can we check our account balance using an ATM?
3. Can we set a PIN using an ATM?

### ATM processing

The interviewer would expect you to ask a question regarding the processing of transactions using an ATM. Therefore, you may ask the following questions:

1. What happens when the amount entered by the user for withdrawal is greater than the user's account balance?
2. What happens when the amount entered by the user for withdrawal is greater than the ATM's cash limit?
3. What happens when the amount entered by the user exceeds the total cash present in the ATM?
4. Can the ATM be used for online transactions?

## Design approach

We are going to design this ATM system using the bottom-up design approach. For this purpose, we will follow the steps below:

* Identify and design the smallest components first—the screen, keypad, cash dispenser, printer, and card reader.
* Use these small components to design bigger ATM components—the state, machine, and room.
* Repeat the steps above until we design the whole ATM system.

## Design patterns

It is always a good practice to discuss the design patterns under which the ATM system falls, during the interview. Stating the design patterns will give the interviewer a positive impression and shows that the interviewee is well-versed in the advanced concepts of object-oriented design.

The following design patterns are used to design the ATM system:

* The Singleton design pattern
* The State design pattern

Let's explore the requirements of the ATM system in the next lesson.

Back

**Requirements for the ATM System**

Learn about all requirements of the ATM design.

**We'll cover the following**

* [Requirement collection](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Requirement-collection)

In this lesson, we’ll list the requirements of the ATM design. This is a very crucial step, since requirements define the scope of a problem. Therefore, getting them right from the interviewer and understanding them well will make the design of the rest of the system smooth and easy.

We’ll use the notational convention to identify each requirement with a unique label, "Rn" where "R" is short for Requirement, and "n" is a natural number.

**Requirement collection**

**R1**: Each user has a single account at the bank that they can access by inserting their card into the ATM.

**R2**: The main components of the ATM system that facilitate interactions between the user and the machine are listed below:

* **Card reader:** To read the user's ATM card
* **Keypad:** To enter information such as the user's PIN
* **Screen:** To display messages to the user, such as prompts or error messages
* **Cash dispenser:** To dispense cash to the user
* **Printer:** To print receipts for the user
* **Network infrastructure:** To connect with the bank's computer system in order to access account information and complete transactions

**R3:**The ATM system must authenticate the user based on the PIN they enter to ensure that only authorized users can access their accounts.

**R4**: All transactions are possible after the successful authentication of the ATM card.

**R5**: The user can have two types of accounts—current and savings—and can perform the following operations on the ATM:

* Balance inquiry
* Cash withdrawal
* Funds/money transfer

**R6**: At the end of a transaction, the user has the option to start another transaction or end their session.

ATM components

We've identified our requirements for the problem. Let’s define different use cases of the ATM system in the next lesson.

# Use Case Diagram for the ATM System

Learn how to define use cases and create the corresponding use case diagram for the ATM.

**We'll cover the following**

* [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
* [Actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actors)
  + [Primary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Primary-actors)
  + [Secondary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Secondary-actors)
* [Use Cases](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-Cases)
  + [Cardholder](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Cardholder)
  + [Card issuer](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Card-issuer)
  + [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
  + [Operator](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Operator)
* [Relationships](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationships)
  + [Associations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Associations)
  + [Extend](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Extend)
  + [Include](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Include)
* [Use case diagram](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-case-diagram)

Let’s build the use case diagram of the ATM and understand the relationship between its different components.

First, we’ll define the different elements of our ATM design, followed by the complete use case diagram of the system.

## System

Our system is an "ATM."

## Actors

Let’s define the main actors of our ATM system.

### Primary actors

**Cardholder:**This is the primary actor of the ATM who inserts or removes the ATM card, performs or cancels the transaction, and changes the PIN.

### Secondary actors

* **Card issuer:** This actor verifies the cardholder's identity, checks for sufficient funds, checks the cardholder's account transaction limit, and blocks/unblocks the cardholder's account.
* **System:** This actor checks for sufficient funds in the ATM, checks the ATM’s transaction limit, and verifies the cardholder's identity. This actor also returns the card and dispenses the amount and receipts.
* **Operator:** The operator is responsible for starting and shutting down the system. It can refill printer receipts and cash dispensers.

## Use Cases

In this section, we will define the use cases for the ATM. We have listed the use cases according to their respective interactions with a particular actor.

### Cardholder

* **Insert card:** To insert an ATM card into ATM
* **Transaction:** To perform an operation—balance inquiry, deposit, fund transfer, or cash withdrawal
* **Change pin:** To change the PIN of the ATM card
* **Cancel transaction:**To cancel a transaction

### Card issuer

* **Verifying the cardholder's identity:**To validate the card and cardholder's bank account details
* **Check sufficient funds in account:** To check if the cardholder's bank account has more or equal funds than the fund being withdrawn or transferred
* **Check account transaction limits:**To check if the transaction limits of the cardholder's bank account are more or equal to the funds being withdrawn or transferred
* **Block/Unblock account:**To block or unblock the cardholder's bank account

### System

* **Verifying the cardholder's identity:**To validate the card and cardholder's bank account details
* **Check sufficient funds in ATM:**To check if the ATM has more or equal funds than the fund being withdrawn or transferred
* **Check ATM withdrawal limits:**To check if the transaction limits of the ATM are more or equal to the funds being withdrawn or transferred
* **Return card:**To return the card after completing or canceling the transaction
* **Dispense money:** To dispense cash in case of cash withdrawal from an ATM
* **Dispense receipt:** To dispense cash after completing the transaction

### Operator

* **System startup/shutdown:** To start and shutdown the ATM session
* **Refill printer receipts:** To refill the paper in the printer
* **Refill cash dispenser:** To refill the cash in the dispenser

## Relationships

This section describes the relationships between and among actors as well as their use cases.

GeneralizationThe “Transaction” use case has a generalization relationship with the “Balance Inquiry,” “Deposit,” “Transfer,” and “Cash withdrawal,” because a cardholder can use any of these options to perform a transaction.

### Associations

The table below shows the association relationship between actors and their use cases.

|  |  |  |  |
| --- | --- | --- | --- |
| **Cardholder** | **Card issuer** | **System** | **Operator** |
| Change PIN | Check for sufficient funds in account | verifying the cardholder's identity | System startup/ shutdown |
| Insert card | Check cardholder account transection limits | Check ATM withdraw limits | Refill printer receipts |
| Cancel transaction | Block/Unblock account | Check for sufficient funds in ATM | Refill cash dispenser |
| Transaction | Verify the cardholder's identity | Return card |  |
|  |  | Dispense money |
| Dispense receipt |

### Extend

The “Transaction” use case has an extend relationship with the “Dispense receipt” use case because we have the option to get a receipt in the case of transactions.

### Include

* When the cardholder inserts the card into the ATM, both the bank and card issuer verify the card and the cardholder's account. Therefore, the “Insert card” use case has an include relationship with the “Verifying the cardholder's identity” use case.
* When the cardholder performs a fund transfer from an ATM, the card issuer verifies two things—if the cardholder's bank account has sufficient funds and if the amount being transferred is within its account's transaction limits. Therefore, the “Transfer” use case has an include relationship with both “Check sufficient funds in account” and “Check account transaction limits.”
* When the cardholder withdraws cash from an ATM, the card issuer verifies two things—if the cardholder's bank account has sufficient funds and if the amount being withdrawn is within its account's transaction limits. The bank also verifies two things—if the ATM has sufficient funds and if the amount being withdrawn is within ATM's withdrawal limits. Therefore, the “Cash withdrawal” use case has an include relationship with “Check sufficient funds in the account,” “Check account transaction limits,” “Check sufficient funds in ATM,” and “Check ATM withdrawal limits.”
* The last cash withdrawal process is the system depositing the money. Therefore, the “Cash Withdrawal” use case has an include relationship with the “Dispense money’ use case.
* When a transaction is performed or canceled, the ATM card is ejected by the ATM. Therefore, both the “Transaction” use case and the “Cancel Transaction” use case have an include relationship with the “Return card” use case.

## Use case diagram

Here’s the use case diagram of the ATM design:

A diagram of a bank

Description automatically generated

# Class Diagram for the ATM System

Learn to create a class diagram for the ATM design using the bottom-up approach.

**We'll cover the following**

* [Components of the ATM system](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Components-of-the-ATM-system)
  + [User](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#User)
  + [ATM card](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ATM-card)
  + [Bank account](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Bank-account)
  + [Bank](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Bank)
  + [Card reader, cash dispenser, keypad, screen, and printer](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Card-reader,-cash-dispenser,-keypad,-screen,-and-printer)
  + [ATM state](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ATM-state)
  + [ATM](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ATM)
  + [ATM room](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ATM-room)
  + [Enumerations and custom data types](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations-and-custom-data-types)
* [Relationship between the classes](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationship-between-the-classes)
  + [Association](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Association)
  + [Composition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Composition)
  + [Inheritance](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Inheritance)
* [Class diagram for the ATM System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Class-diagram-for-the-ATM-System)
* [Design pattern](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-pattern)
* [Additional requirements](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Additional-requirements)

In this lesson, we’ll design the classes and then identify the relationship between classes according to the requirements for the ATM design problem.

## Components of the ATM system[#](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Components-of-the-ATM-system)

As mentioned earlier, we’ll design the class diagram for the ATM using a bottom-up approach.

### User

The User class represents a user with an ATM card and a bank account.

A screenshot of a computer

Description automatically generatedBankAccount is a parent class with two types: SavingAccount and CurrentAccount. These classes are derived from the BankAccount class. This class stores the account number, total balance, and the user's available balance.

* SavingAccount: This derived class represents a saving account with a withdrawal limit.
* CurrentAccount: This derived class represents a current/checking account with a withdrawal limit.

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

### ATM state

ATMState is an abstract class with six types: CheckBalanaceState, CashWithdrawalState, TransferMoneyState, HasCardState, IdleState, and SelectOperationState. These classes are derived from the ATMState class. This class decides the state of the ATM system and several states including the return card and exit of the ATM system.

* CheckBalanceState: This class represents the state that allows users to check their account balance.
* CashWithdrawalState: This class represents the state that allows users to withdraw cash.
* TransferMoneyState: This class represents the state that allows users to transfer money.
* HasCardState: This class represents the state that checks whether or not the user has a valid card and authenticates the card’s PIN.
* IdleState: This class represents the state where the ATM system is idle and is not performing any functions.
* SelectOperationState: This class represents the state that allows users to select an operation for the ATM to perform.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A diagram of a bank

Description automatically generated

### Inheritance

The following classes show an inheritance relationship:

* Both, SavingAccount and CurrentAccount, extend the BankAccount class.
* The CheckBalanceState, CashWithdrawalState, TransferMoneyState, HasCardState, IdleState, and SelectOperationState classes extend the abstract class, ATMState.

**Note:** We have already discussed the inheritance relationship between classes in the component section above one by one.

## Class diagram for the ATM System

Here’s the complete class diagram for our ATM design:

A diagram of a machine

Description automatically generated

## Design pattern

The following design patterns have been used in the class diagram:

* **The Singleton design pattern:** This pattern ensures the existence of a single instance of the ATM at a given moment that can be accessed by multiple users, due to the shared nature of the ATM components.
* **The State design pattern:** This pattern enables the ATM to alter its behavior based on the internal changes in the machine. This way, an ATM can transition from one state to another, like switching from an idle state to displaying an account balance or money withdrawal state, and as soon as all the operations have been performed, it can switch back to the initial idle state.

The following design patterns can also be used to design ATM:

* The Composite design pattern can be used to combine different components of the ATM along with their functionalities.
* The Builder design pattern allows the same processes for a complex object to have different representations. In the ATM system, it can help separate different kinds of transactions like withdrawals, deposits, etc.

## Additional requirements

There is a chance that the interviewer might ask about the working of the cash withdrawal process. How can it be implemented in our ATM system? This addition is a bit challenging since we need a system that can withdraw the correct combinations of hundred, twenty, and two dollar bills, respectively, according to the amount specified by the user. The system also needs to work sequentially until the required amount is met.

We will use the Chain of Responsibility design pattern to tackle this addition to our system. This design pattern will ensure the correct division of the dollar bills in the ATM by creating a chain of handlers that forward the requests based on the situation until all the requirements are met.  
We have created the following classes to implement the Chain of Responsibility design pattern:

* CashWithdrawProcessor: This is associated with the CashWithdrawalState class. This abstract class is extended by HundredDollarWithdrawProcessor, TwentyDollarWithdrawProcessor, and TwoDollarWithdrawProcessor.
* HundredDollarWithdrawProcessor: This class is derived from CashWithdrawProcessor and is responsible for withdrawing hundred-dollar bills based on the requirement.
* TwentyDollarWithdrawProcessor: This class is derived from CashWithdrawProcessor and is responsible for withdrawing twenty-dollar bills based on the requirement.
* TwoDollarWithdrawProcessor: This class is derived from CashWithdrawProcessor and is responsible for withdrawing two-dollar bills based on the requirement.

**Valid Amount:**If the amount entered by the user has a modulus equal to zero with any of the specified bills that the ATM can withdraw, then the amount is considered valid for the transaction. If the amount is invalid, then the transaction will not be processed.

For example, a user wants to withdraw $548. The HundredDollarWithdrawProcessor class will start the cash withdrawal using the cashWithdrawal() method by taking out five bills of hundred dollars. Now that we have $48 to withdraw for the user which is less than a hundred dollars, the TwentyDollarWithdrawProcessor class will start withdrawing dollar bills. This class will take out two bills of twenty dollars with $8 remaining. Since two dollars is less than twenty, the cashWithdrawal() method of the TwoDollarWithdrawProcessor will take out four bills of $2 for the user. The withdrawal, in this case, is successful

A screenshot of a computer screen

Description automatically generated

**Sequence Diagram for the ATM System**

Create a sequence diagram for how to complete the balance inquiry using an ATM.

**We'll cover the following**

* [Balance inquiry](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Balance-inquiry)

A sequence diagram is a great way to understand the interactions between different entities and objects in the system. There can be different sequence diagrams that we can create for our ATM. For the sake of this lesson, we will create sequence diagrams for balance inquiry from ATM.

**Balance inquiry**

The sequence diagram for how to complete the balance inquiry using an ATM should have the following actors and objects that will interact with each other:

* **Actors:** Cardholder, Card issuer
* **Objects:**ATM and Printer

Here are the steps for balance inquiry from ATM:

1. The cardholder inserts their ATM card into the ATM.
2. The ATM then asks the cardholder for their PIN.
3. The cardholder enters their PIN code, and the ATM sends the code to the card issuer for verification.
4. The card issuer verifies the PIN code:
   1. If the PIN is valid, the card issuer sends a message back to the ATM indicating that the verification was successful.
      1. The ATM then displays the main menu to the cardholder.
      2. The cardholder selects the balance inquiry option.
      3. The ATM sends a message to the card issuer to get the account balance, and the card issuer returns the balance to the ATM.
      4. The ATM displays the balance to the cardholder and asks if they would like a receipt printed.
         1. If the cardholder selects "yes," the ATM sends a message to the printer to print a receipt and ejects the ATM card.
         2. If the cardholder does not want a receipt, the ATM card is simply ejected.
   2. If the PIN is not verified, and the ATM card is simply ejected.

Based on the order above, the sequence diagram of balance inquiry from ATM is given below.

A screenshot of a computer screen

Description automatically generated

A screenshot of a diagram

Description automatically generated

# Activity Diagram for the ATM System

Let’s create some activity diagrams for the ATM.

**We'll cover the following**

* [The cardholder performs ATM transactions](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#The-cardholder-performs-ATM-transactions)
  + [States](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#States)
  + [Actions](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actions)
* [Activity challenge: The cardholder performs an ATM cash withdrawal](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Activity-challenge:-The-cardholder-performs-an-ATM-cash-withdrawal)

An activity diagram is a great way to visualize the flow of messages from one activity to the other in the system. There can be different activity diagrams that we can create for our ATM. In this lesson, we will create activity diagrams for the following two activities:

* The cardholder performs ATM transactions.
* **Activity challenge:**The cardholder performs an ATM cash withdrawal.

## The cardholder performs ATM transactions

The followings are the states and actions that will be involved in this activity diagram.

### States

**Initial state:**The cardholder inserts an ATM card.

**Final state:**There are two final states in this activity diagram as shown below:

* The cardholder gets an error message for the wrong ATM PIN.
* The cardholder performs a transaction successfully.

### Actions

The cardholder approaches the ATM, inserts their card, and enters their PIN. If the PIN is correct, they are allowed to perform a transaction. If the PIN is incorrect, the ATM returns the card. The cardholder then has the option to select from four different transactions. After completing their chosen transaction, the cardholder receives their card back from the ATM.

Based on the order above, the activity diagram is given below.

A diagram of a card

Description automatically generated

## Activity challenge: The cardholder performs an ATM cash withdrawal

You’ll help us create an activity diagram of a cardholder performing an ATM cash withdrawal.

The skeleton of the activity diagram is shown below.

A diagram of a card payment method

Description automatically generated

Notice that the actions in the diagram above are numbered from 1 to 6. The slots shown below represent the activities, and the arrows represent the flow from one activity to the other. Can you rearrange the slots below in the correct order they should appear in the activity diagram above?

Alternatively, click the "Show complete diagram" button below to see the complete sequence diagram:

A diagram of a card

Description automatically generatedWe've looked at some of the activity diagrams of our ATM. In the next lesson, we will present the code for our designed classes in some of the most popular languages.

# Code for the ATM System

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

**We'll cover the following**

* [ATM classes](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ATM-classes)
  + [Enumerations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations)
  + [User and ATM card](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#User-and-ATM-card)
  + [Bank and bank account](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Bank-and-bank-account)
  + [Card reader, card dispenser, printer, screen, and keypad](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Card-reader,-card-dispenser,-printer,-screen,-and-keypad)
  + [ATM state](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ATM-state)
  + [ATM and ATM room](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#ATM-and-ATM-room)
* [Wrapping up](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Wrapping-up)

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.

ATMState: 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.

Java

// Enumeration

enum ATMState {

Idle,

HasCard,

SelestionOption,

Withdraw,

TransferMoney,

BalanceInquiry

}

A screenshot of a computer

Description automatically generated

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();

}

### 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:

Java

C#

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();

}

### 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 returnCard() {

// definition

}

@Override

public void exit(ATM atm) {

// definition

}

}

public class HasCardState extends ATMState {

@Override

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

// definition

}

@Override

public void returnCard() {

// definition

}

@Override

public void exit(ATM atm) {

// definition

}

}

public class SelectOperationState extends ATMState {

@Override

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

// definition

}

@Override

public void returnCard() {

// definition

}

@Override

public void exit(ATM atm) {

// definition

}

}

public class CheckBalanceState extends ATMState {

@Override

public void displayBalance(ATM atm, ATMCard card) {

// definition

}

@Override

public void returnCard() {

// definition

}

@Override

public void exit(ATM atm) {

// definition

}

}

public class CashWithdrawalState extends ATMState {

@Override

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

// definition

}

@Override

public void returnCard() {

// definition

}

@Override

public void exit(ATM atm) {

// definition

}

}

public class TransferMoneyState extends ATMState {

@Override

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

// definition

}

@Override

public void returnCard() {

// definition

}

@Override

public void exit(ATM atm) {

// definition

}

}

### 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;

public void displayCurrentState();

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

}

public class ATMRoom {

private ATM atm;

private User user;

}

**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.