Object-Oriented Static Modeling of the Banking System - III

Lecture # 32

Problem Description

- A bank has several automated teller machines (ATMs), which are geographically distributed and connected via a wide area network to a central server. Each ATM machine has a card reader, a cash dispenser, a keyboard/display, and a receipt printer. By using the ATM machine, a customer can withdraw cash from either checking or savings account, query the balance of an account, or transfer funds from one account to another. A transaction is initiated when a customer inserts an ATM card into the card reader. Encoded on the magnetic strip on the back of the ATM card are the card number, the start date, and the expiration date. Assuming the card is recognized, the system validates the ATM card to determine that the expiration date has not passed, that the user-entered PIN (personal identification number) matches the PIN maintained by the system, and that the card is not lost or stolen. The customer is allowed three attempts to enter the correct PIN; the card is confiscated if the third attempt fails. Cards that have been reported lost or stolen are also confiscated.

Problem Description

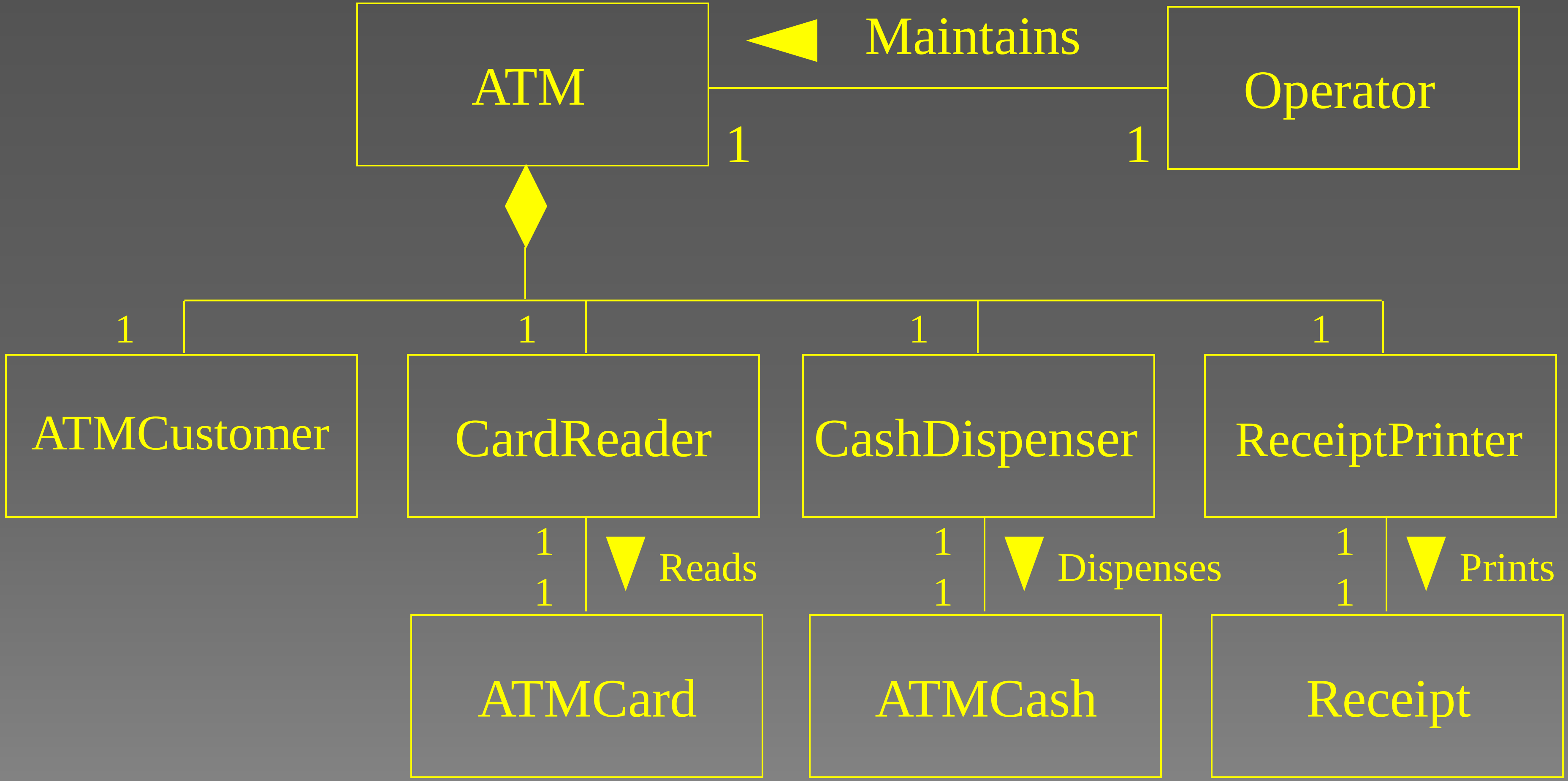
- If the PIN is validated satisfactorily, the customer is prompted for a withdrawal, query, or transfer transaction. Before withdrawal transaction can be approved, the system determines that sufficient funds exist in the requested account, that the maximum daily limit will not be exceeded, and that there are sufficient funds available at the local cash dispenser. If the transaction is approved, the requested amount of cash is dispensed, a receipt is printed containing information about the transaction, and the card is ejected. Before a transfer transaction can be approved, the system determines that the customer has at least two accounts and that there are sufficient funds in the account to be debited. For approved query and transfer requests, a receipt is printed and card ejected. A customer may cancel a transaction at any time; the transaction is terminated and the card is ejected. Customer records, account records, and debit card records are all maintained at the server.

Problem Description

- An ATM operator may start up and close down the ATM to replenish the ATM cash dispenser and for routine maintenance. It is assumed that functionality to open and close accounts and to create, update, and delete customer and debit card records is provided by an existing system and is not part of this problem.

Putting the Classes Together

Conceptual Static Model for Problem Domain: Physical Classes



Conceptual Static Model for Problem Domain: Physical Classes

- A bank has several ATMs.
- Each ATM is a composite class consisting of a Card Reader, a Cash Dispenser, a Receipt Printer, and a user, the ATM Customer, who interacts with the system by using a keyboard/display

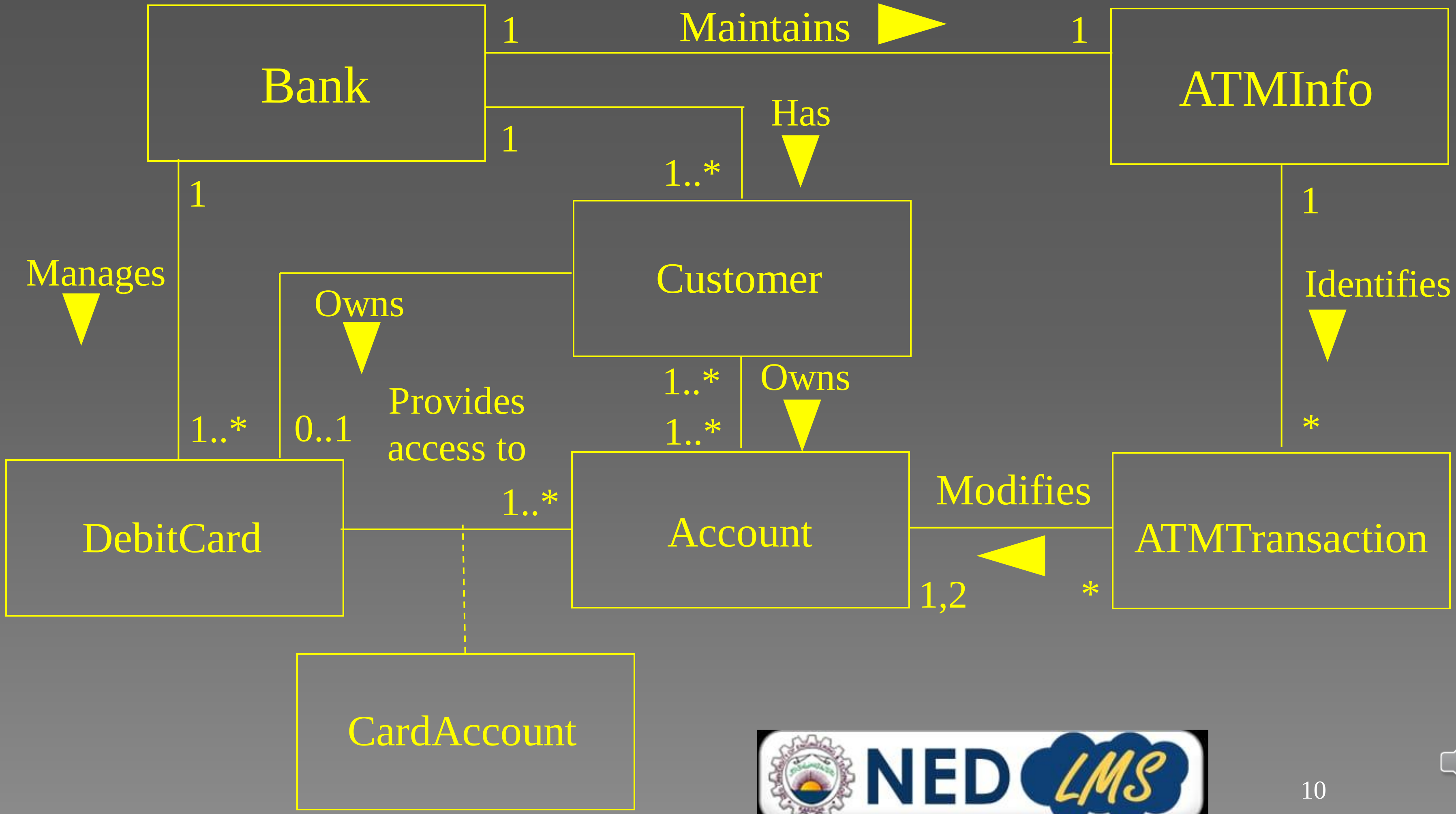
Conceptual Static Model for Problem Domain: Physical Classes

- The Card Reader reads an ATM Card, which is a plastic card and hence a physical entity
- The Cash Dispenser dispenses ATM Cash, which is also a physical entity in terms of paper money of given denominations

Conceptual Static Model for Problem Domain: Physical Classes

- The Receipt Printer prints a Receipt, which is a paper physical entity
- The physical entities represent classes in the problem domain and are usually modeled by software entity classes
- In addition, the Operator is also a an external user, whose job is to maintain the ATM – who interacts with the system via a keyboard/displaytemuuteat

Conceptual Static Model for Problem Domain: Entity Classes



Conceptual Static Model for Problem Domain: Entity Classes

- The Bank entity class has a one-to-many relationship with the Customer class and the Debit Card class
- The Bank class has only one instance
- The Customer has many to many relationship with Account class, which has its subclasses (Checking Account, Savings Account)

Conceptual Static Model for Problem Domain: Entity Classes

- An Account is modified by an ATM Transaction, which is also specialized to depict different types of transactions (Withdrawal Transaction, Query Transaction, Transfer Transaction, or PIN Validation Transaction)

Conceptual Static Model for Problem Domain: Entity Classes

- There is also a Card Account association class. Association classes are needed in cases where the attributes are of the association, rather than of the classes connected by the association. Thus, in many-to-many association between Debit Card and Account, the individual accounts that can be accessed by a given debit card are attributes of the Card Account class and not of either Debit Card or Account

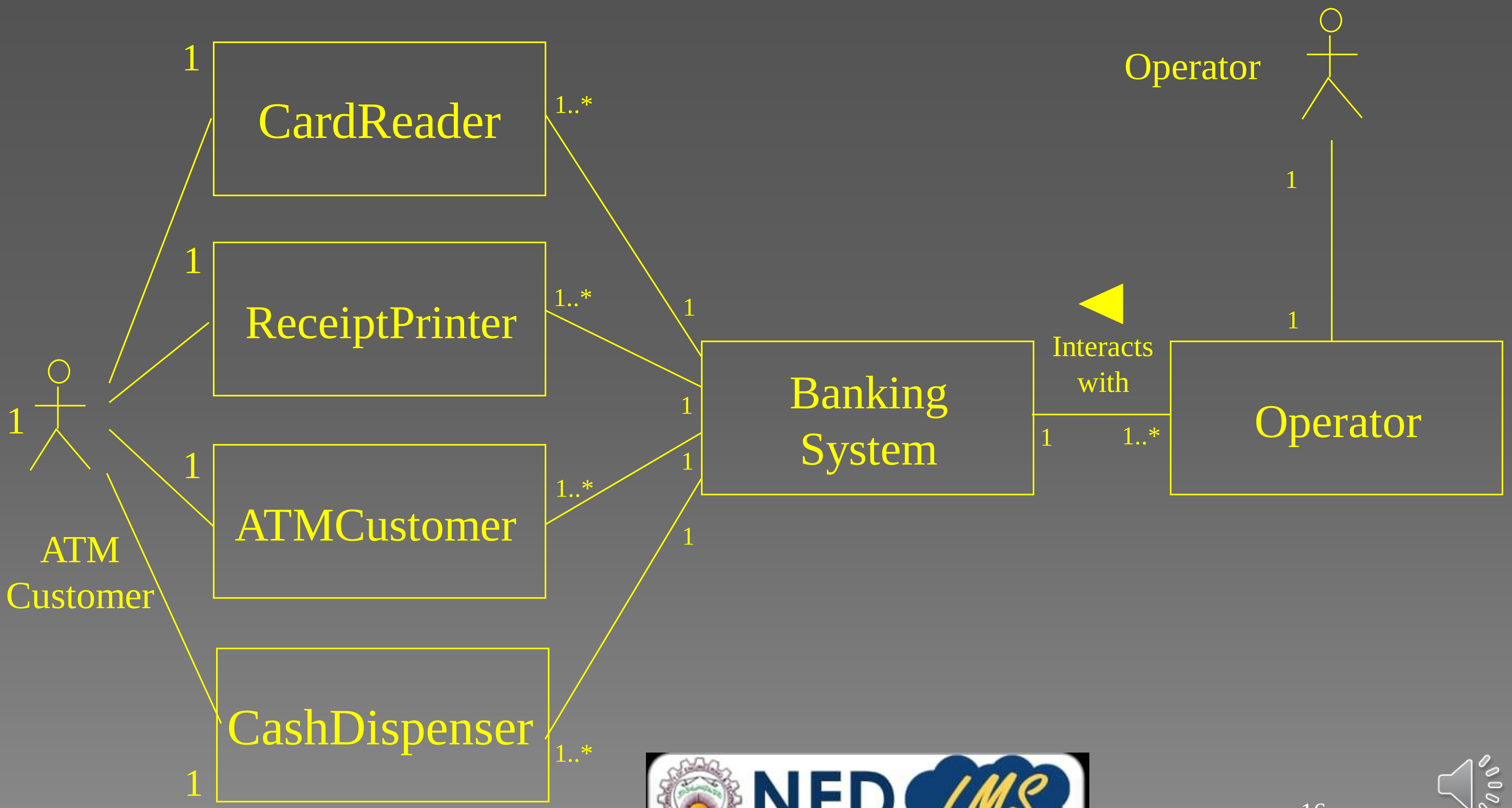
Conceptual Static Model for Problem Domain: Entity Classes

- Entity classes are also required to model the physical classes in the problem domain
- These include ATM Card, representing the information read off the magnetic strip on the plastic card

Conceptual Static Model for Problem Domain: Entity Classes

- ATM Cash holds the amount of cash maintained at an ATM, in five-hundred and one-thousand rupee notes
- The Receipt holds information about a transaction, and because is holds similar information to the Transaction class described earlier, a separate entity class is unnecessary

Banking System Context Class Diagram



Banking System Context Class Diagram

- The system context diagram is developed to show the external classes to which the Banking system has to interface
- We develop the context class diagram by considering the physical classes

Banking System Context Class Diagram

- The external classes correspond to the users and I/O devices depicted to the classes in the application domain
- They are the Card Reader, the Cash Dispenser, the Receipt Printer, the ATM Customer who interacts with the system via a keyboard/display, and the Operator who also interacts with the system via a keyboard/display

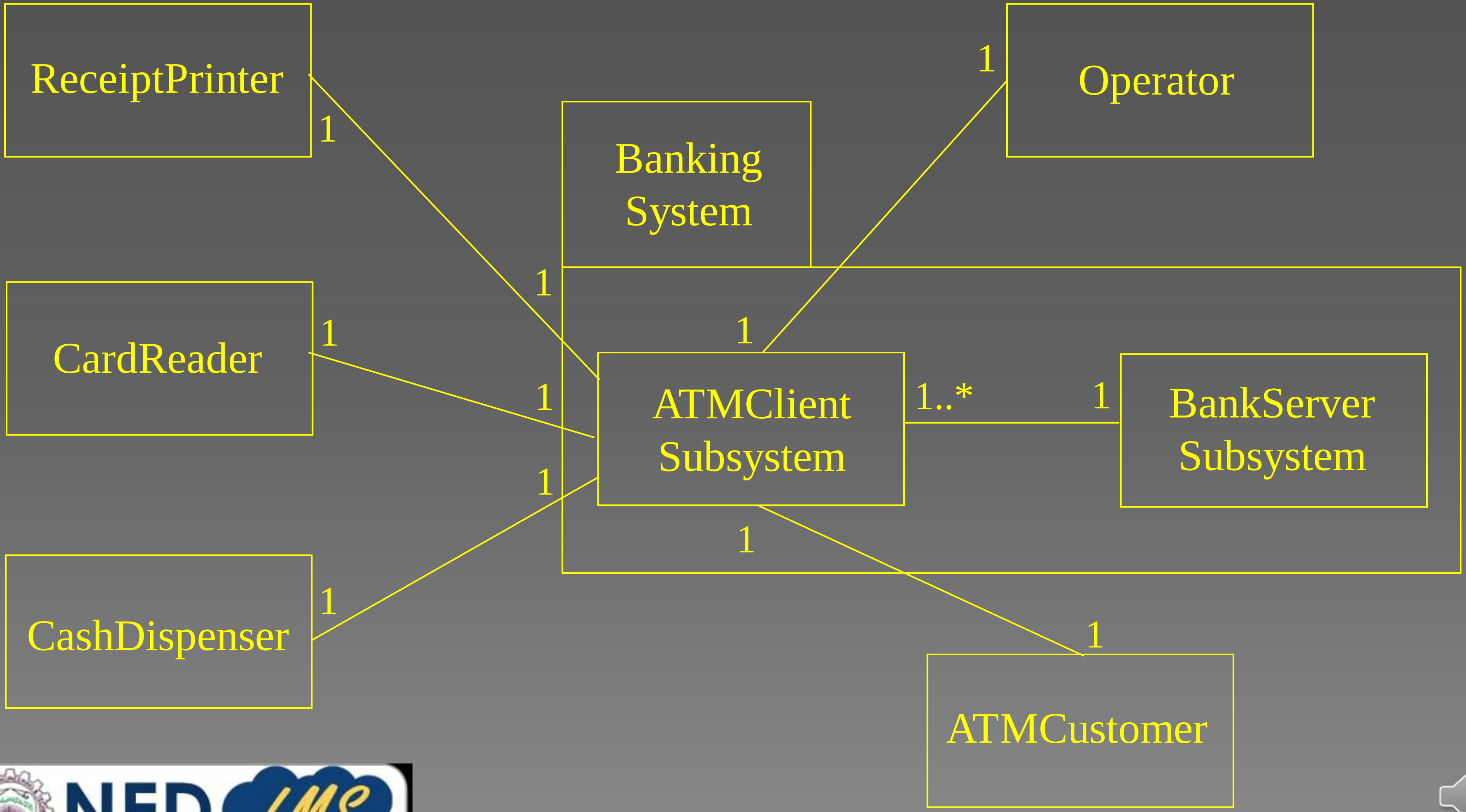
Banking System Context Class Diagram

- There is one instance of each of these external classes for each ATM
- The system context class diagram for the Banking system depicts the system as one aggregate class that receives input from and provides output to the external classes

Client/Server Subsystem Structuring

- The Banking system is a client/server application, some of the objects reside at the ATM client and some reside at the bank server
- The ATM Client Subsystem
- The Bank Server Subsystem

Banking System: Major Subsystems



Banking System: Major Subsystems

- The ATM Client Subsystem is a composite class, and one instance of this class is located at each ATM machine
- All the external classes interface to this aggregate class

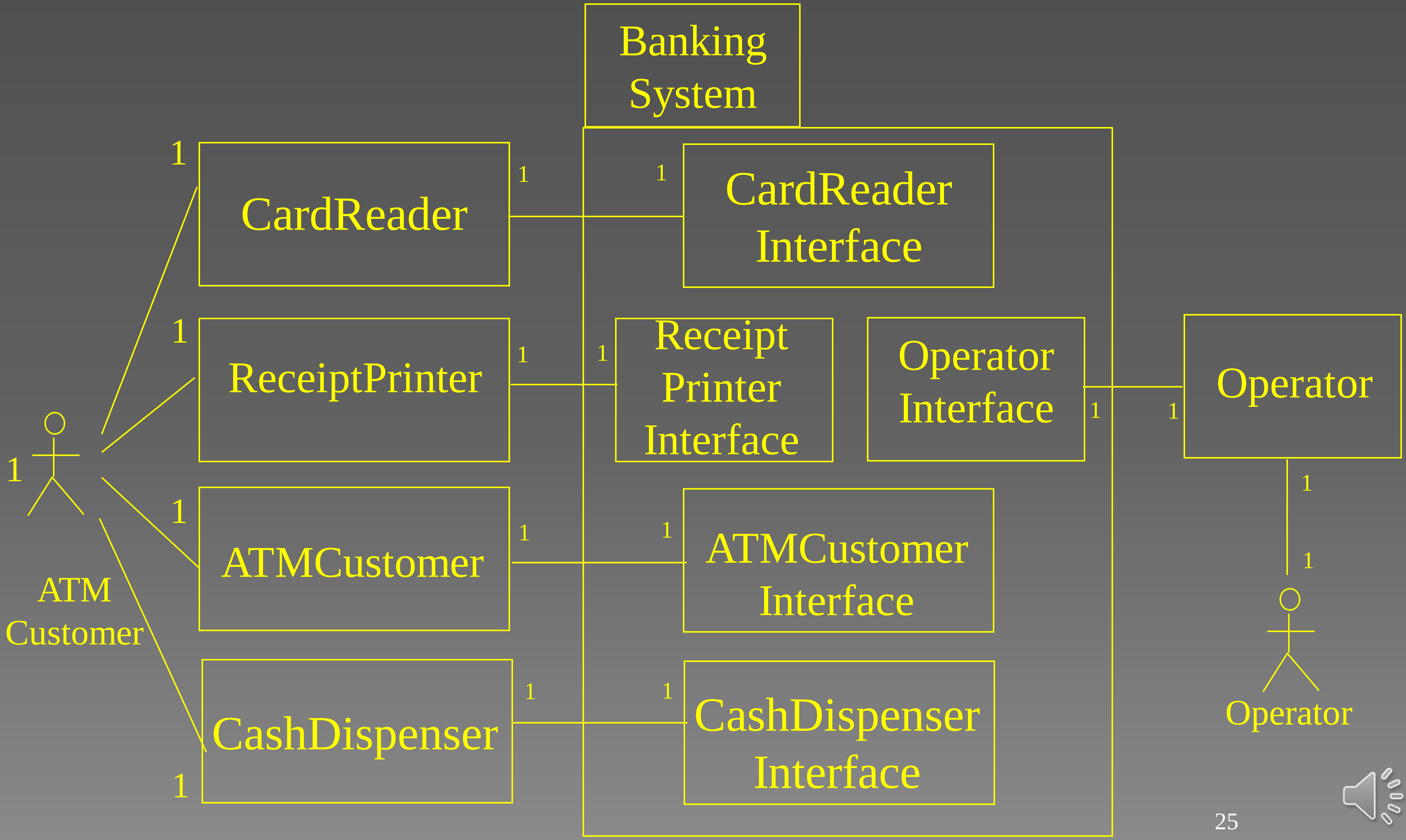
Banking System: Major Subsystems

- The Bank Server Subsystem has one instance
- This aggregate class has a one-to-many association with the ATM Client Subsystem
- This is an example of geographically distributed system

ATM Client Object Structuring

- Consider the interface classes for the ATM Client Subsystem. These classes are determined from the system context diagram
- We design one interface class for each external class, which can be a device interface class or user interface class

Banking System External Classes and Interfaces Classes



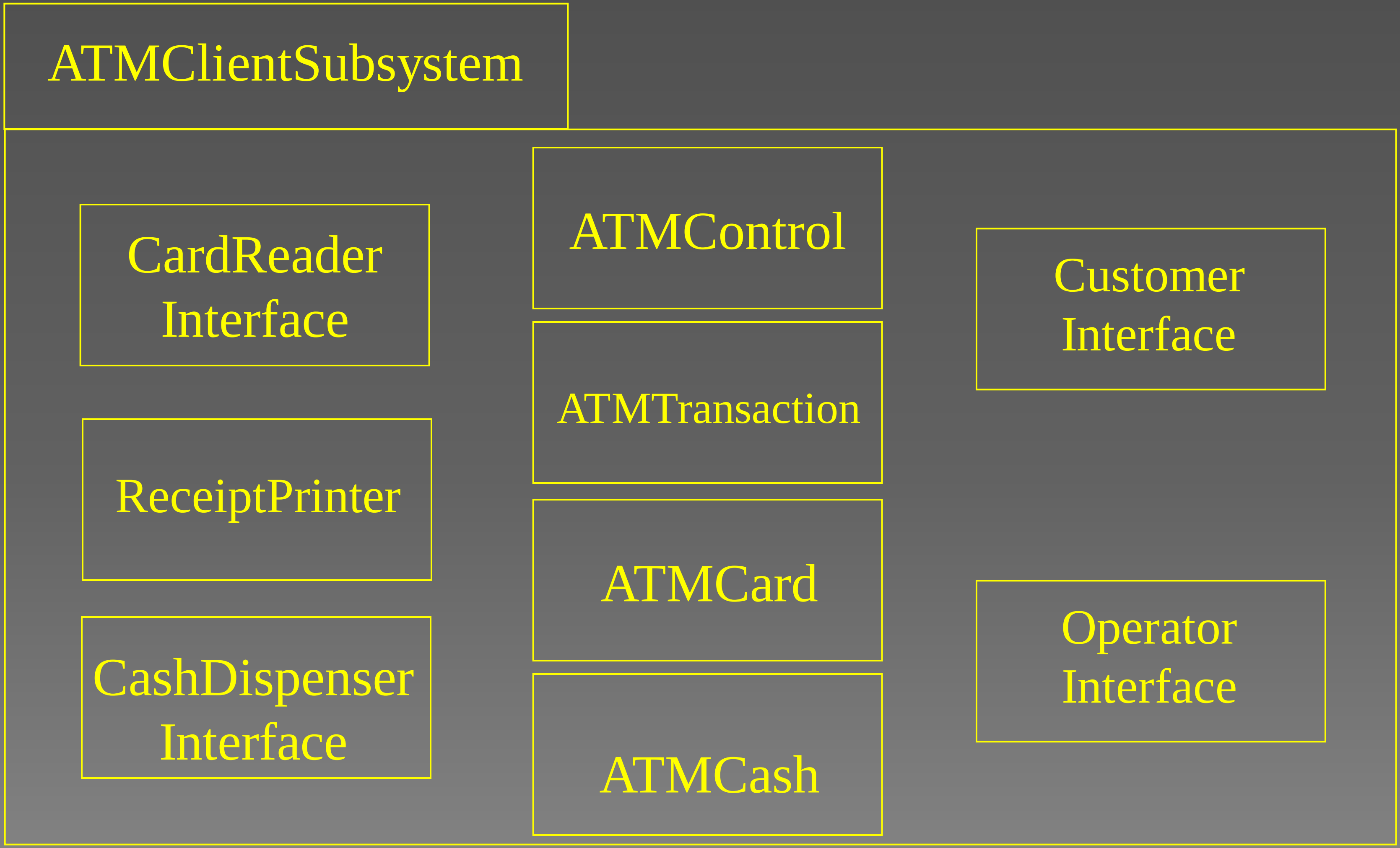
Banking System External Classes and Interfaces Classes

- The device interface classes are the Card Reader Interface, through which ATM cards are read, the Cash Dispenser Interface, which dispenses cash, and the Receipt Printer Interface, which prints receipts

Banking System External Classes and Interfaces Classes

- There is also the Customer Interface, which is the user interface class that interacts with the customer via the keyboard/display, displaying textual messages, prompting the customer, and receiving the customer’s inputs
- The Operator Interface class provides the user interface to the ATM operator, who replenishes the ATM machine with cash
- There is one instance of each of these interface classes for each ATM

ATM Client Subsystem Classes



The ATM Control Class

- To control the sequence in which actions take place, we identify the need for a control object/class
- This is a state-dependent object

ATM Client Structuring: Objects in Use Cases

Validate PIN Abstract Use Case

- Card Reader Interface
- ATM Card
- Customer Interface
- ATM Transaction (PIN Validation Transaction)
- ATM Control controls the sequence

Withdraw Funds Concrete Use Case

- ATM Transaction (Withdrawal Transaction)
- Cash Dispenser Interface
- ATM Cash
- Receipt Printer Interface
- ATM Control controls the sequence

Use Cases and Classes

- Query Account and Transfer Funds use cases are associated with the same set of classes/objects

Operator-Specific Use Cases

- Operator Interface
- ATM Control

Object Structuring in Bank Server Subsystem

- Several entity objects are bank-wide and need to be stored to be accessible from any ATM. Therefore, these objects must be stored at the server
- These objects include: Account objects and Debit Card objects

Object Structuring in Bank Server Subsystem

- In the Bank Server Subsystem, the entity classes are Customer, the Account super-class, Checking Account and Savings Account subclasses, and Debit Card

Object Structuring in Bank Server Subsystem

- There is also the ATM Transaction object, which migrates from the client to the server. The client sends the transaction request to the server, which sends a response to the client. The transaction is also stored at the server as an entity in the form of a Transaction Log, so that a transaction history is maintained

Object Structuring in Bank Server Subsystem

- The transient data sent as part of the ATM Transaction message might differ from the persistent transaction data; for example transaction status is known at the end of the transaction but not during it

Object Structuring in Bank Server Subsystem

- Business logic objects are also needed at the server to define the business-specific application logic for processing client requests. Each ATM transaction type needs a transaction manager to specify the business rules for handling the transaction

Summary

- Continued the discussion of Banking System Case Study.
- Performed the static modeling of different classes and objects which exists in the case study domain