Software Engineering-2

ATM OO Design and Implementation Case Study

By:

Dr. Salwa Osama

Dr. Ahmed Hesham

ATM

- Popular example—people can relate to it
- Available in 4 languages
  - Java, C++, Visual Basic and Visual C#
    - “How to Program” books

Case Study Coverage

- Starts with System Requirements
- Presents 6 UML diagram types
  - Use case, class, state machine, activity, communication, sequence
- Simplified design process
  - Requirements document

Design Process

- Examine the Requirements Document
- Identify the Classes
- Identify Class Attributes
- Identify Objects' States and Activities
- Identify Class Operations
- Indicate Collaboration Among Objects
- Start to Program the Classes
- Incorporate Inheritance and Polymorphism into the design

Examining the Requirements

- A local bank intends to install a new ATM to allow bank customers to perform basic financial transactions

- Each customer can have only one account at the bank.

- ATM bank customers

View their account balance

Withdraw cash

Deposit funds

Examining the Requirements

- ATM user interface:

- The cash dispenser begins each day loaded with 500 $20 bills.

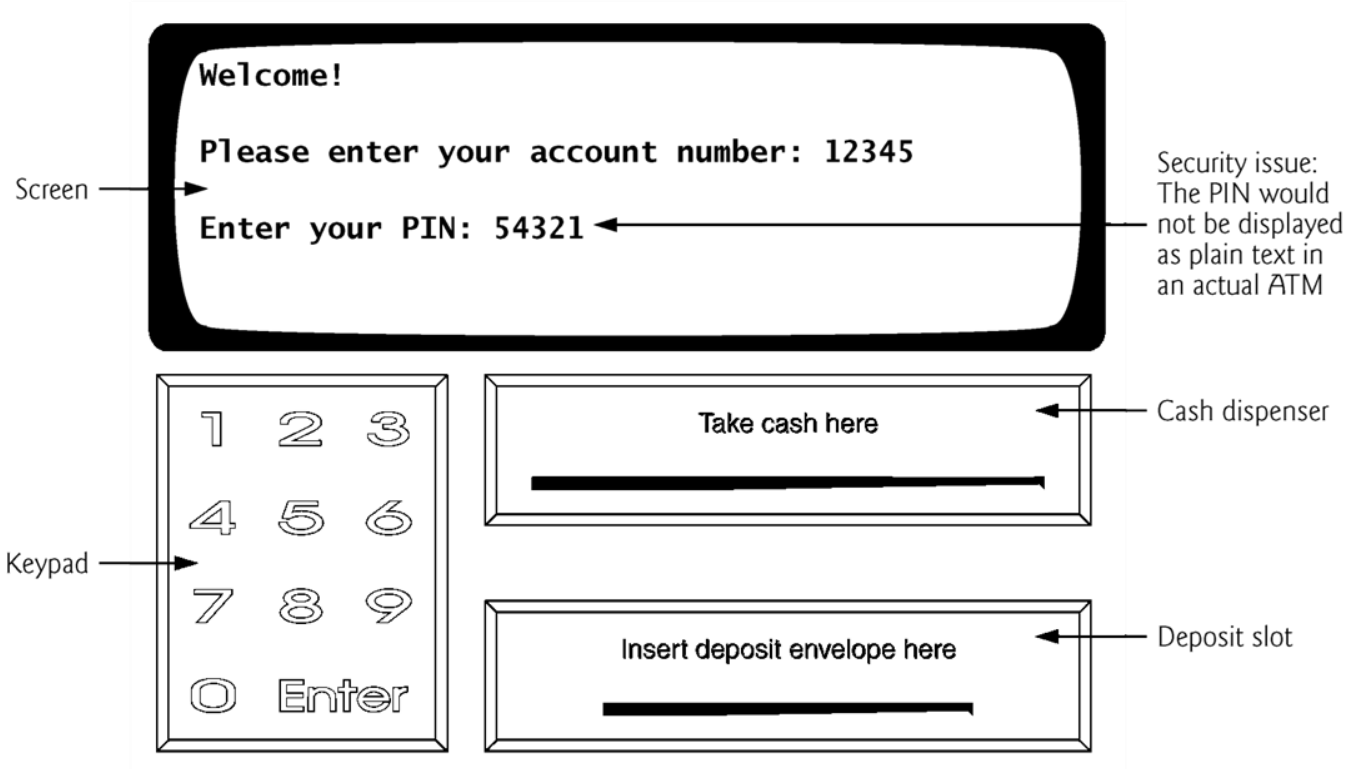


Fig. 12.1 | Automated teller machine user interface.

Examining the Requirements

- Simplifying Assumptions:
  - The bank plans to build only one ATM
  - Each user has only one account
  - Use the computer's **monitor to simulate** the ATM's screen
  - Use the computer's **keyboard to simulate** the ATM's keypad
  - The bank **does not make any changes** to the information **in the database** while a user is accessing the ATM.
  - The bank **trusts** the ATM to **access** and manipulate the information in the **database** without significant security measures.

Examining the Requirements

- Upon first approaching the ATM, the user should experience the following sequence of events:
  1. The screen displays Welcome! and prompts the user to enter an account number.
  2. The user enters a five-digit account number using the keypad.
  3. The screen prompts the user to enter the PIN.
  4. The user enters a five-digit PIN using the keypad.
  5. If the user enters a valid account number and the correct PIN for that account, the screen displays the main menu.
  6. If the user enters an invalid account number or an incorrect PIN, the screen displays an appropriate message, then the ATM returns to *Step 1* to restart the authentication process.

Examining the Requirements

- **If the user enters 1** to make a balance inquiry, the screen displays the user's account balance.
- The ATM must retrieve the balance from the bank's database.

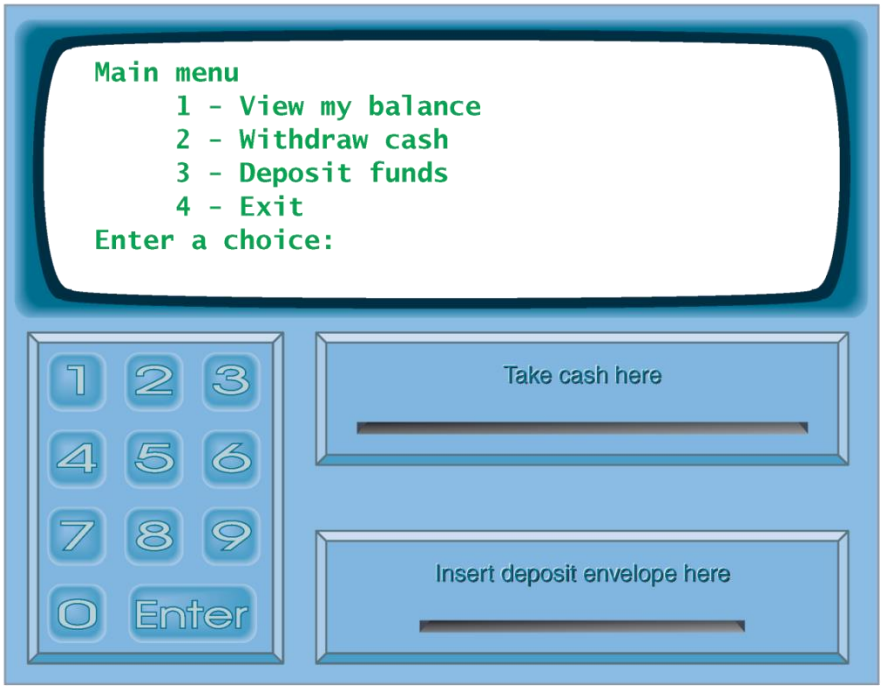


Fig. 12.2 | ATM main menu.

Examining the Requirements

When a user selects 2 for withdrawal, ...

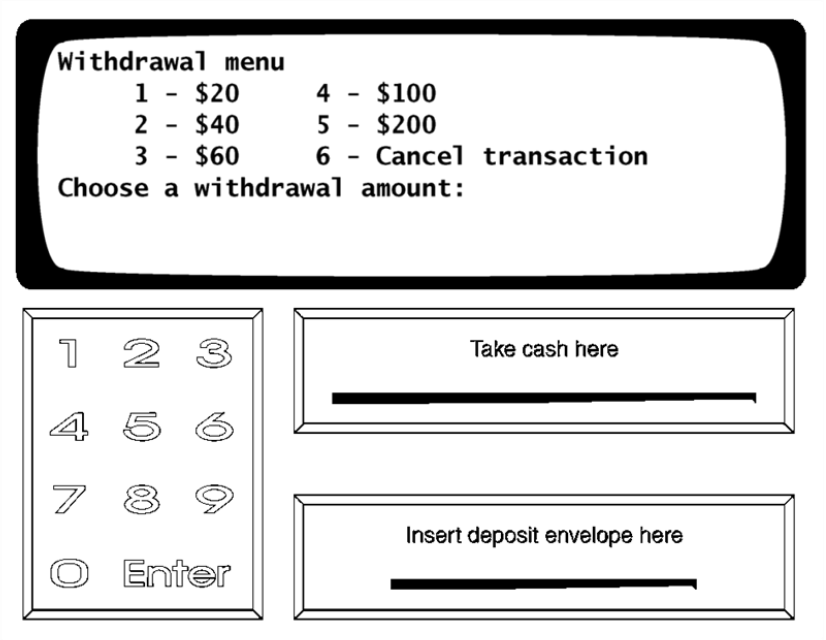


Fig. 12.3 | ATM withdrawal menu.

Examining the Requirements

- The user enters a menu selection using the keypad.
  - If the withdrawal amount is greater than the user's account balance, …
  - If the withdrawal amount is less than or equal to the user's account balance ...
  - If the user chooses to cancel, ...
- If the cash dispenser contains enough cash, ... Otherwise ..
- The ATM debits the withdrawal amount from the user's account in the bank's database.
- The cash dispenser dispenses the desired amount of money.
- The screen displays a message reminding the user to take the money.

Examining the Requirements

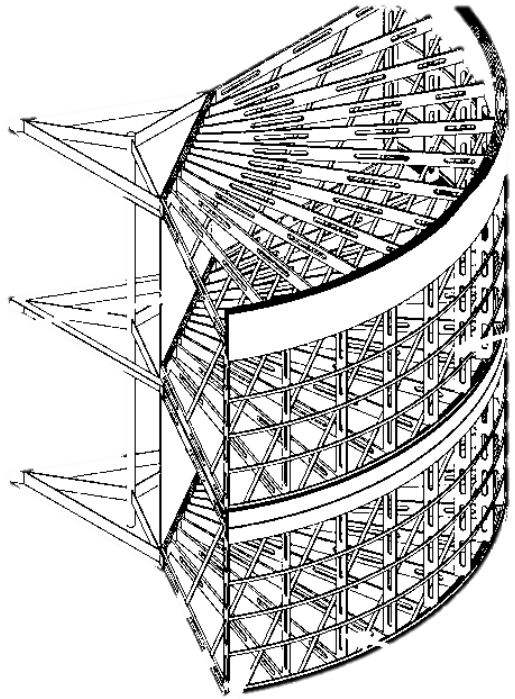
- When the user **enters 3 to make a deposit**:
  - The screen prompts the user to enter a deposit amount or type 0 (zero) to cancel.
    - The user enters a deposit amount or 0 using the keypad.
  - If the user specifies a deposit amount, the ATM proceeds ...
  - If the user chooses to cancel, the ATM ...
  - The screen displays a message telling the user to insert a deposit envelope.
  - If the deposit slot receives a deposit envelope within two minutes, the ATM credits the deposit amount to the user’s account in the bank’s database.

What are UML diagrams?

- By using UML, the entire software design is easier to read and understand prior to software development, thus reducing development risks. Also, it facilitates communication between various developers.
- UML diagrams are divided into structural diagrams and behavioral diagrams.

Structural Diagrams

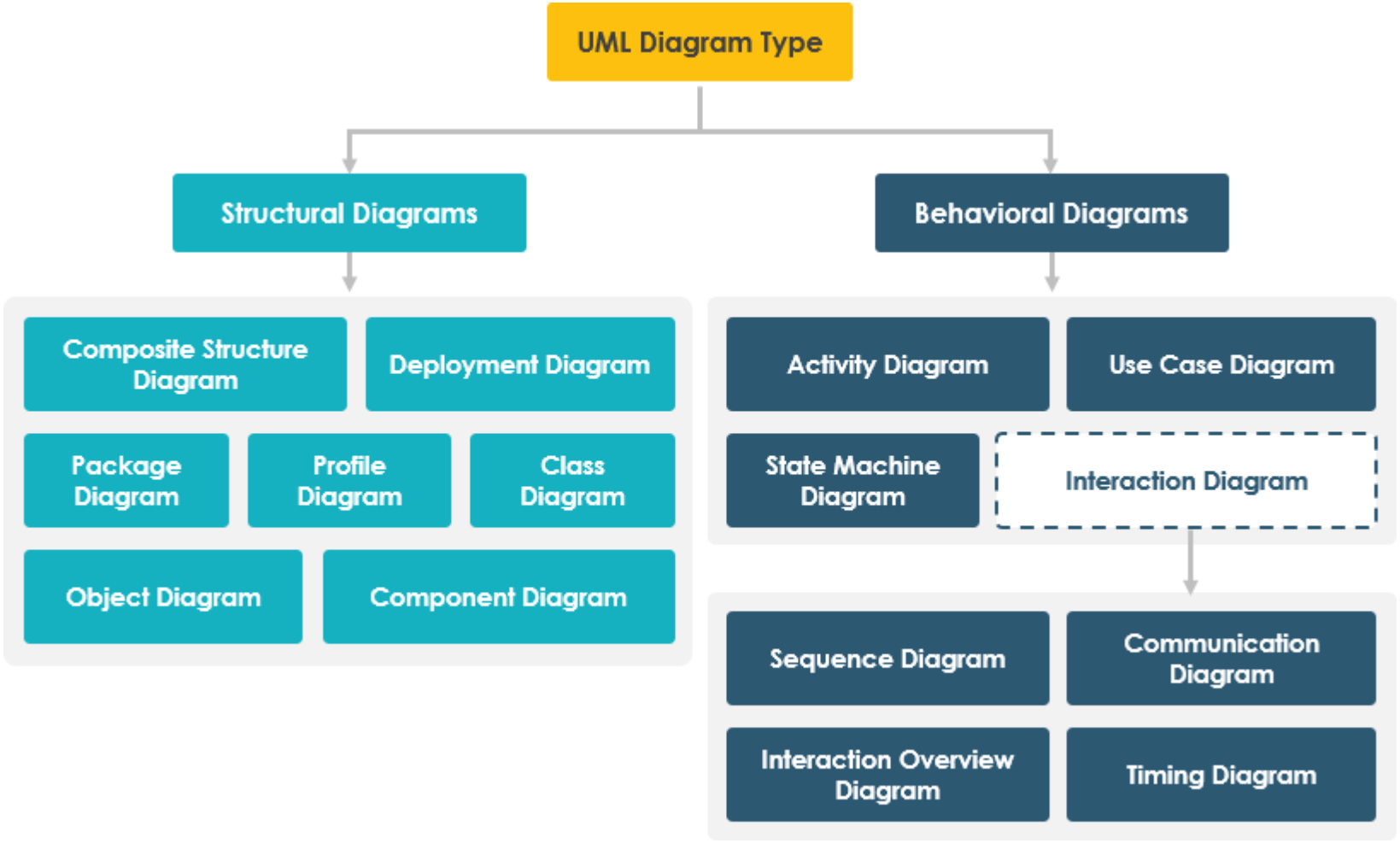
- As the name suggests, they depict the overall structure of a system – its objects, attributes, and relationship between different entities. That is, we are more concerned with what the system is about rather than how it behaves.



Behavioral UML Diagrams

- These diagrams define the behavior of the system and the overall functionality of every unit.
- Also, some of these diagrams depict the flow of information/control in the system as well.

What are UML diagrams?



Analyzing the ATM system

- Interactions between a system and its external entities (actors)

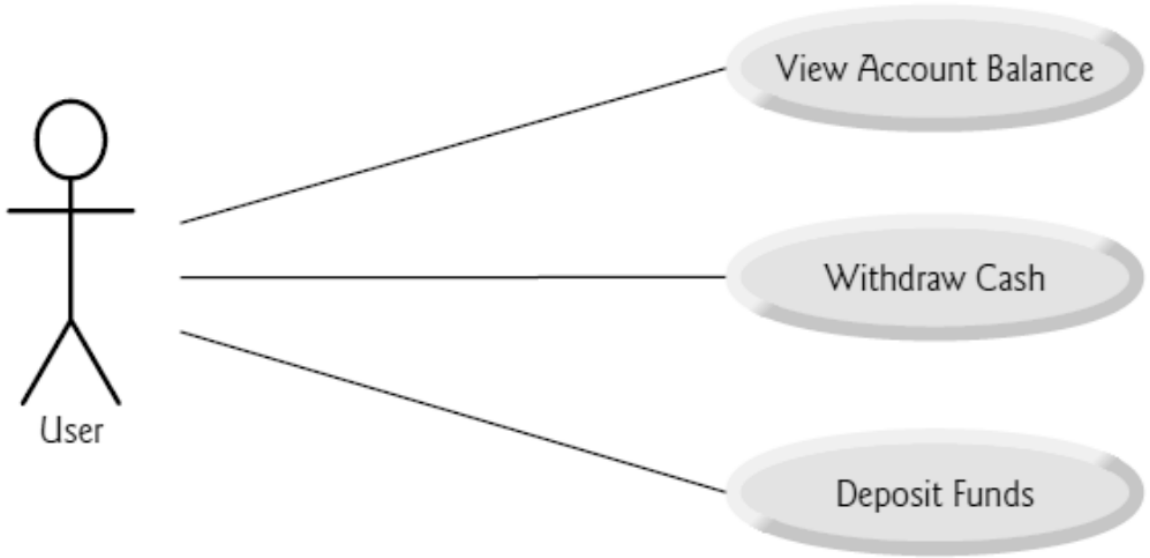


Fig. 12.4 | Use case diagram for the ATM system from the User's perspective.

1-Use Case Diagram

Design the ATM systems

The output of the design stage—a design specification—should specify clearly *how* the system should be constructed to satisfy these requirements.

Identifying the **Classes** in a Requirements

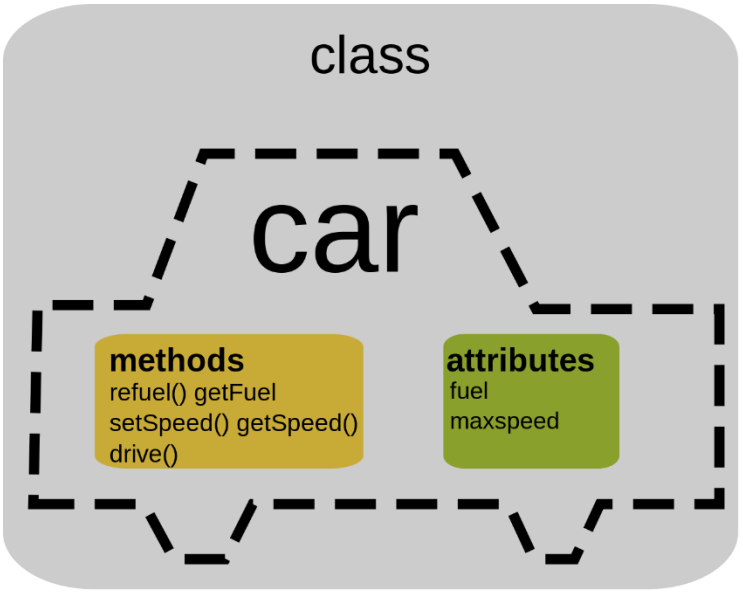
- Analyze the *nouns and noun phrases*
- Create classes only for the nouns and noun phrases that have significance in the ATM system.
- Some of these nouns and noun phrases are actually attributes
- Some of the nouns do not correspond to parts of the system
- Additional classes may become apparent to us as we proceed through the design process.

Nouns and noun phrases in the ATM requirements document			
bank	money / funds	account number	ATM
screen	PIN	user	keypad
bank database	customer	cash dispenser	balance inquiry
transaction	$20 bill / cash	withdrawal	account
deposit slot	deposit	balance	deposit envelope

Fig. 12.5 | Nouns and noun phrases in the ATM requirements document.

Identifying the **Classes** in a Requirements

- Classes:
  - ATM
  - screen
  - keypad
  - cash dispenser
  - deposit slot
  - account
  - bank database
  - balance inquiry
  - withdrawal
  - deposit



Diagrams

- Model classes used in a system.
- Specify structural relationships between system parts

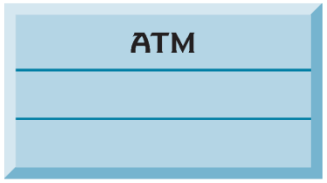


Fig. 12.6 | Representing a class in the UML using a class diagram.

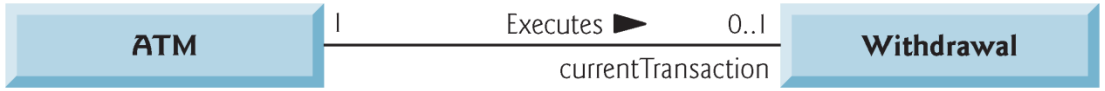
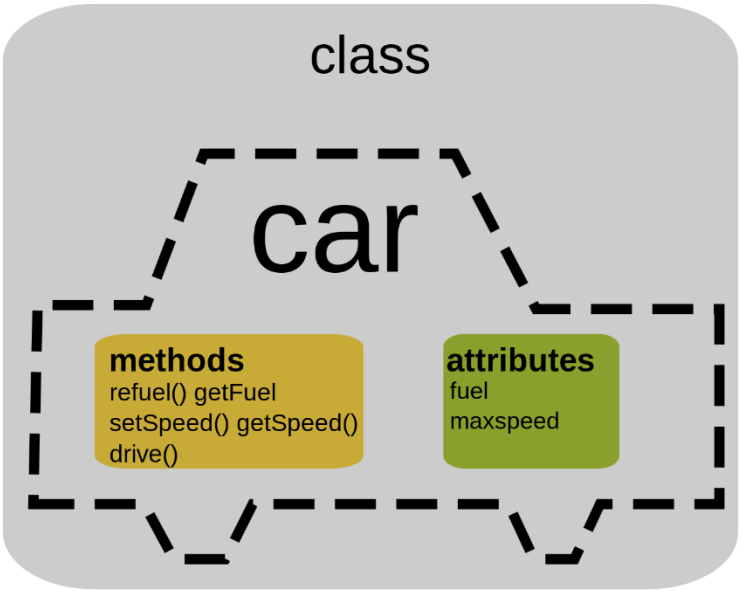


Fig. 12.7 | Class diagram showing an association among classes.

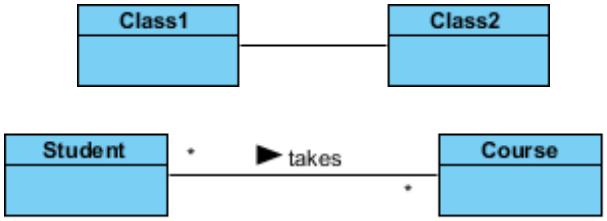
2-Class Diagram

Class Diagram Review

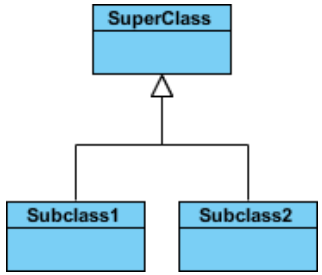
- A class notation consists of three parts:
  1. Class Name
  2. Class Attributes
  3. Class Operations (Methods)
- Class Relationships



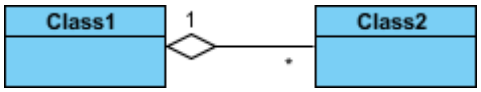
Class Diagram Review - Class Relationships



Simple Association
A structural link between two peer classes.
Represents an "**has-a**" relationship



Inheritance (or Generalization)
Represents an "**is-a**" relationship.



Aggregation
A special type of association.
It represents a "**part of**" relationship

Composition
A special type of aggregation where parts are **destroyed** when the whole is destroyed.

Identifying the **Classes** in a Requirements

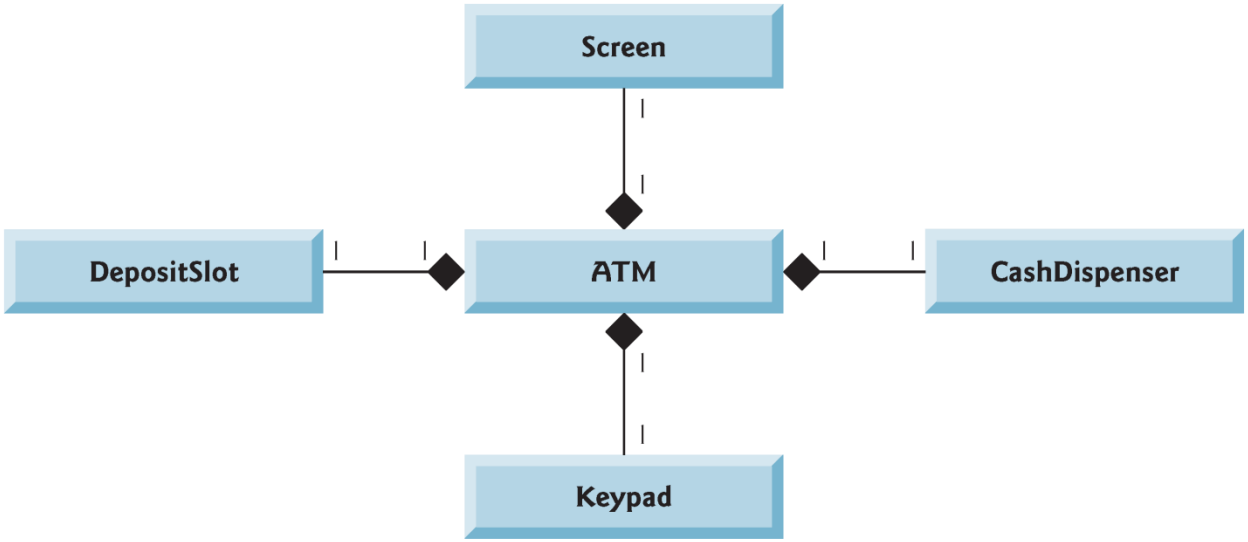


Fig. 12.9 | Class diagram showing composition relationships.

Identifying the **Classes** in a Requirements

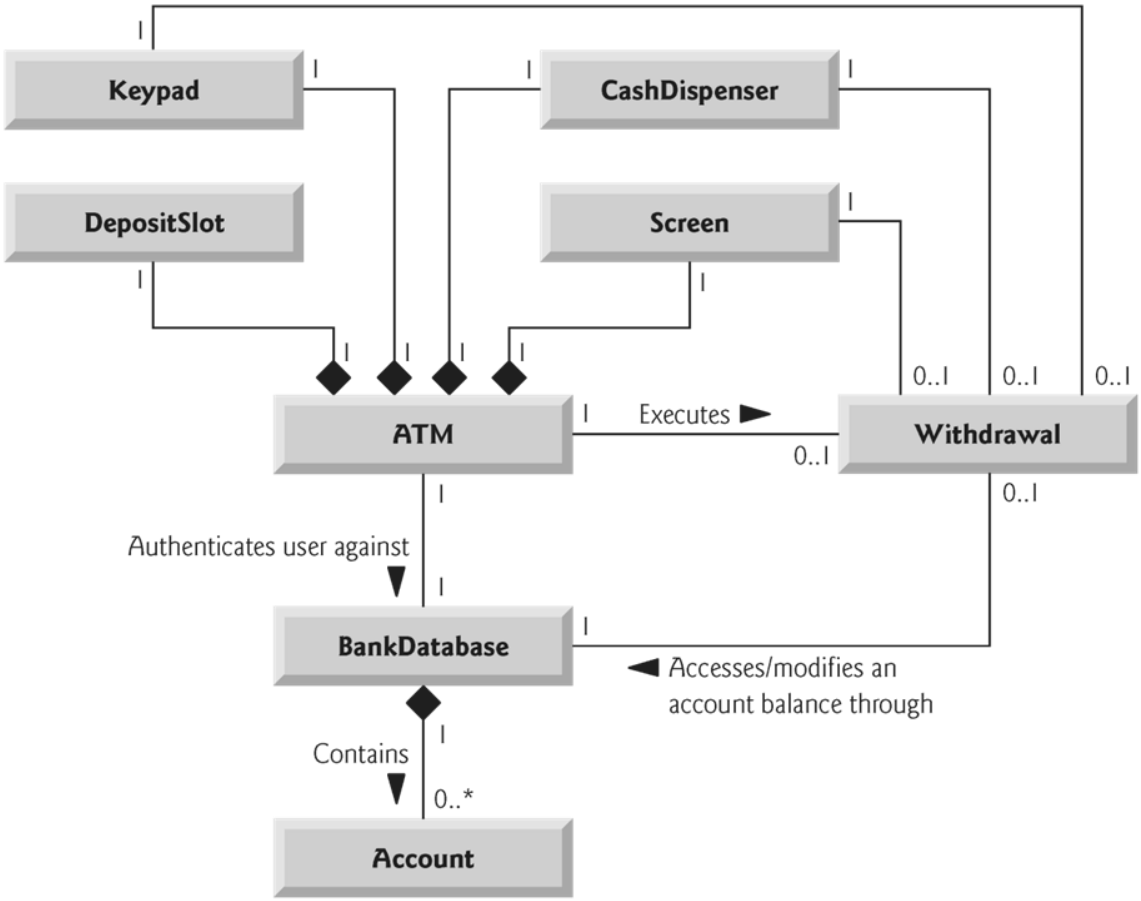


Fig. 12.10 | Class diagram for the ATM system model.

Identifying the **Classes** in a Requirements

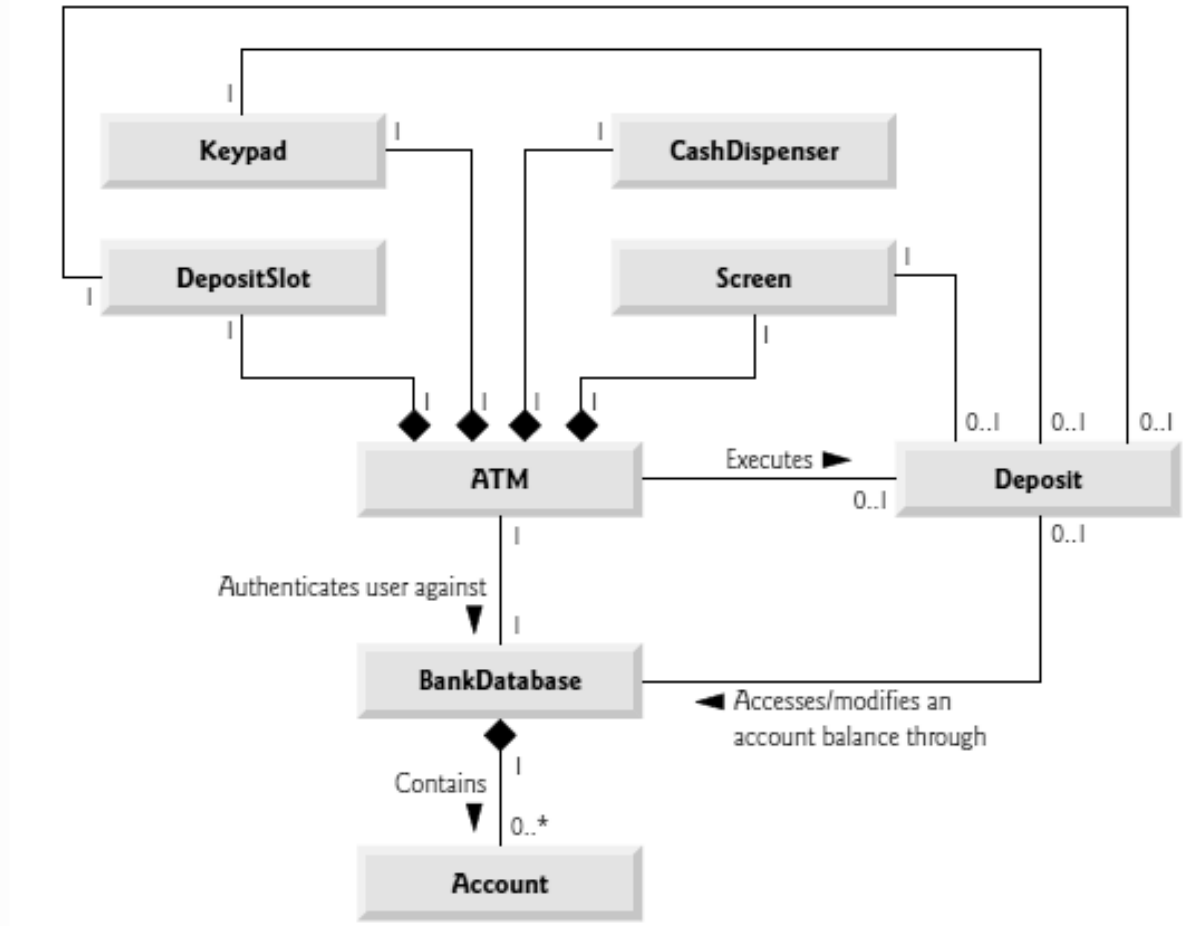


Fig. 12.28 | Class diagram for the ATM system model including class `Deposit`.

Identifying Class Attributes

- Classes have attributes (data) and operations (behaviors).
- Look for descriptive words and phrases in the requirements document.
- For each such word and phrase we find that plays a significant role in the ATM system, we create an attribute and assign it to one or more of the classes

Identifying Class Attributes

Class	Descriptive words and phrases
`ATM`	user is authenticated
`BalanceInquiry`	account number
`Withdrawal`	account number amount
`Deposit`	account number amount
`BankDatabase`	*[no descriptive words or phrases]*
`Account`	account number PIN balance
`Screen`	*[no descriptive words or phrases]*
`Keypad`	*[no descriptive words or phrases]*
`CashDispenser`	begins each day loaded with 500 $20 bills
`DepositSlot`	*[no descriptive words or phrases]*

Fig. 12.11 | Descriptive words and phrases from the ATM requirements document.

Identifying Class Attributes

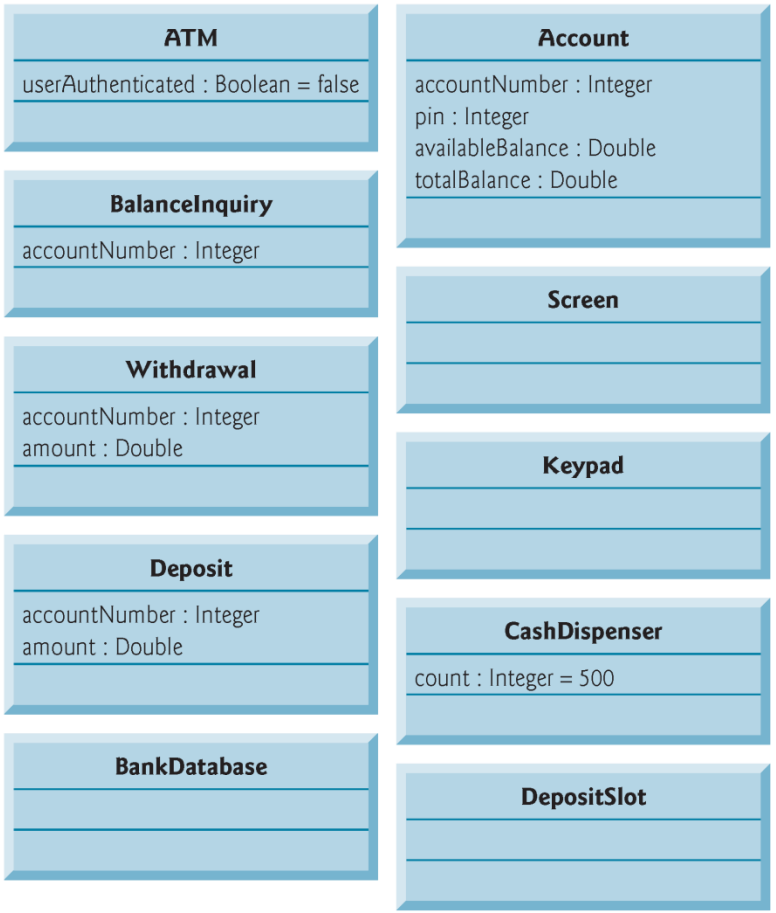


Fig. 12.12 | Classes with attributes.

Software Engineering Observation 12.1

At early stages in the design process, classes often lack attributes (and operations). Such classes should not be eliminated, however, because attributes (and operations) may become evident in the later phases of design and implementation.

Identifying Objects' States and Activities

- After adding attributes to classes, **a state diagram** can be used to show how those attributes represent an object's state and how the state of an object changes.
- An **activity diagram** can be used to model the actions (sequence of events) the object will perform and in what order.

Diagrams

- Model how an object changes state
- Indicated by the values of object attributes at a given time

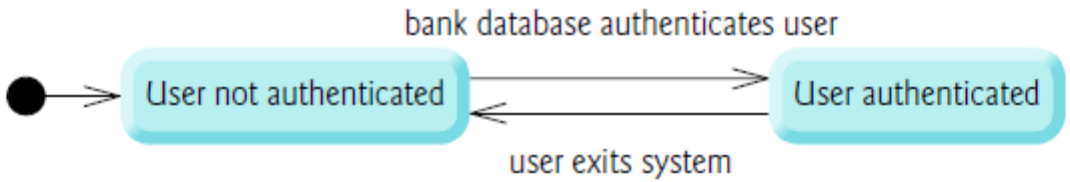


Fig. 12.13 | State diagram for the ATM object.

3-State Diagram

Software Engineering Observation 12.2

Software designers do not generally create state diagrams showing every possible state and state transition for all attributes—there are simply too many of them. State diagrams typically show only key states and state transitions.

Diagrams

- Model workflow during program execution
- Model actions and order in which they're performed

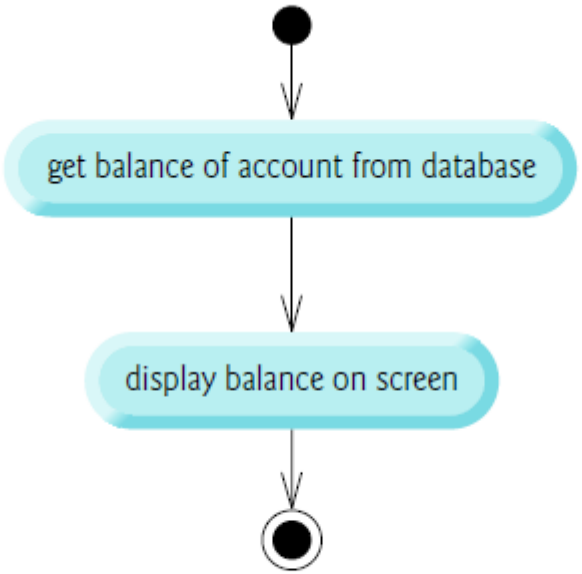


Fig. 12.14 | Activity diagram for a `BalanceInquiry` object.

4-Activity Diagram

Withdrawal transaction

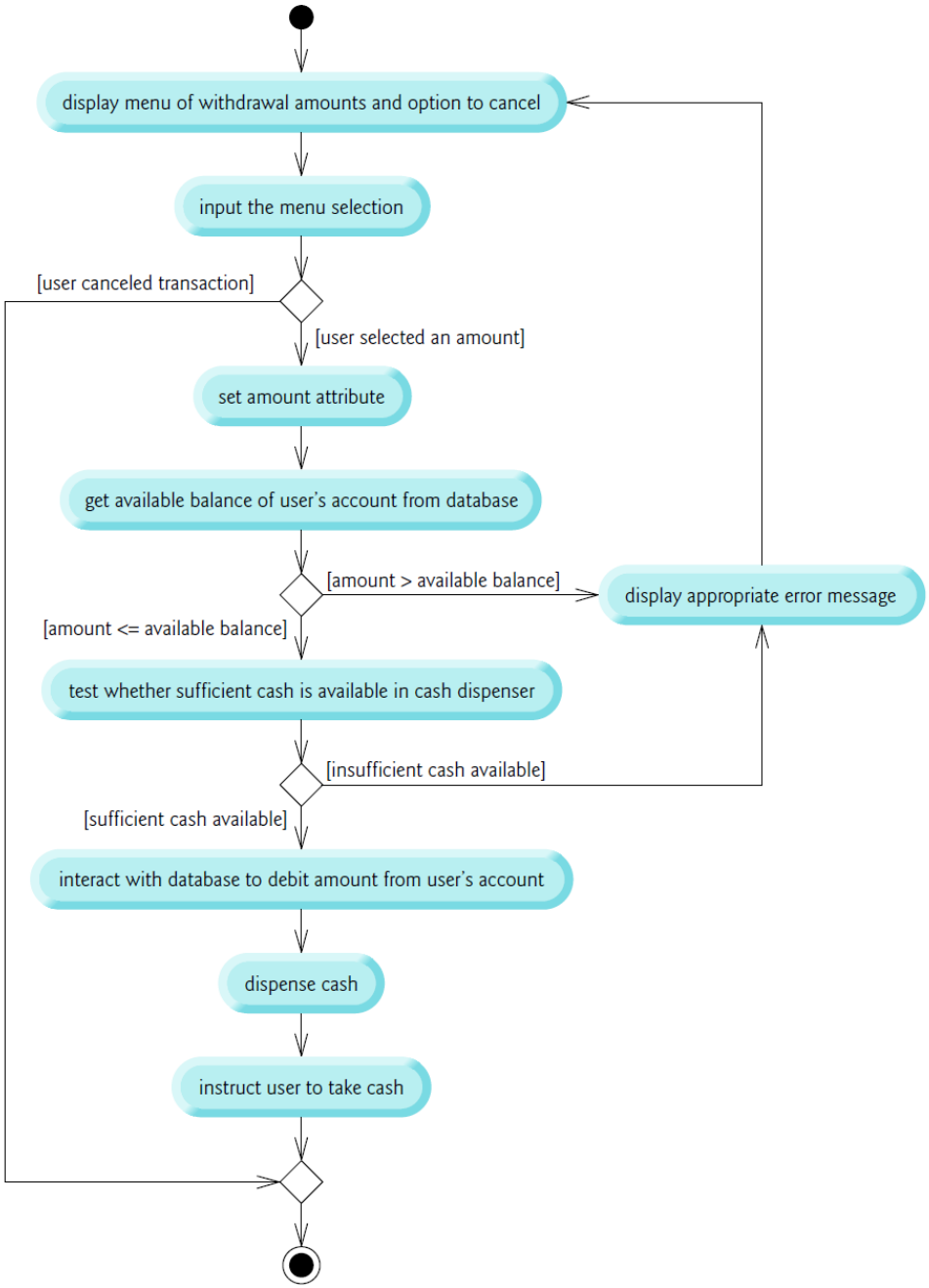


Fig. 12.15 | Activity diagram for a withdrawal transaction.

Deposit transaction

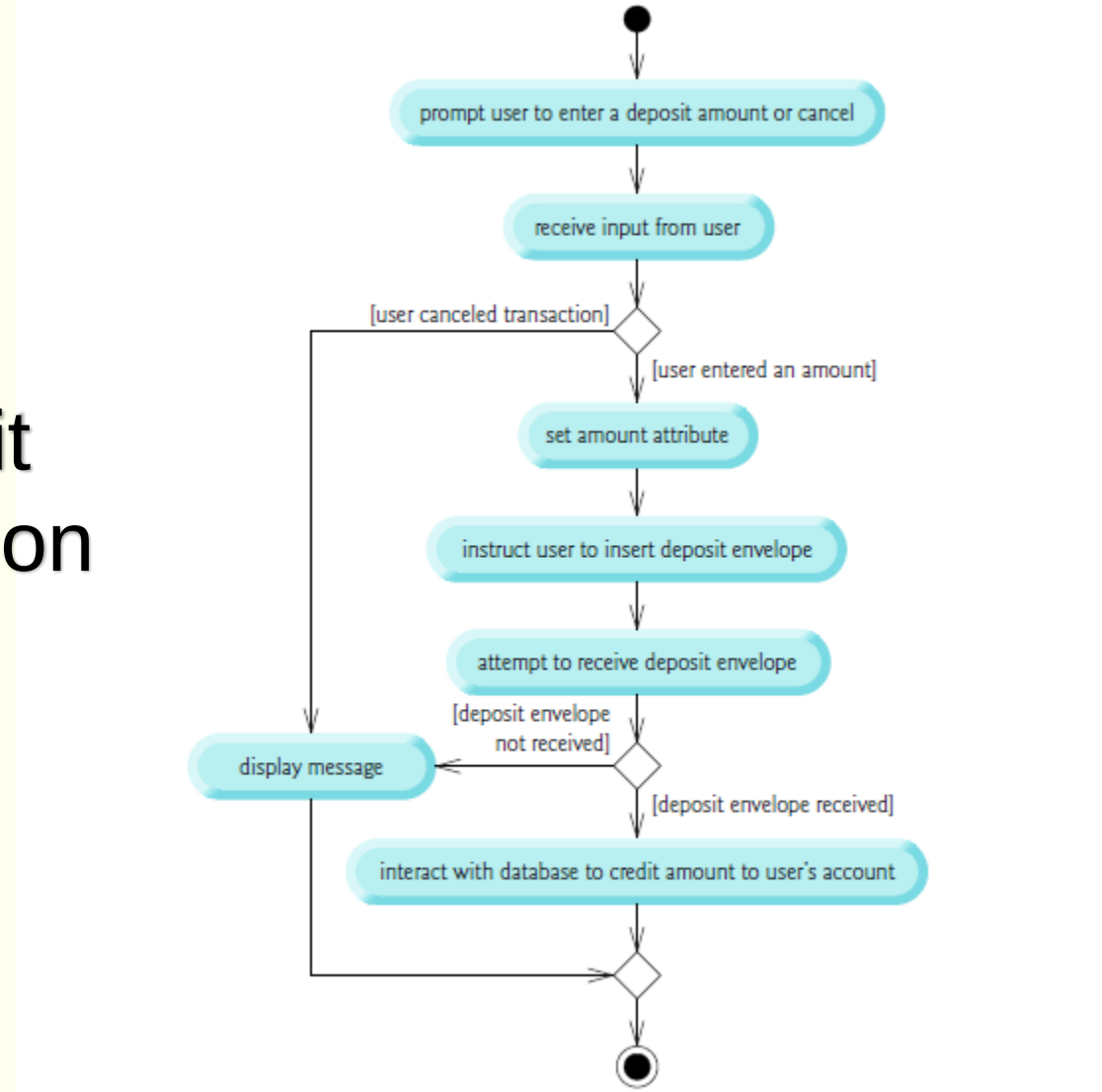


Fig. 12.29 | Activity diagram for a deposit transaction.

Identifying Class Operations

- An operation is a service that objects of a class provide to clients (users) of the class.
- We can derive many of the class operations by examining the key verbs and verb phrases in the requirements document.

Class	Verbs and verb phrases
`ATM`	executes financial transactions
`BalanceInquiry`	*[none in the requirements document]*
`Withdrawal`	*[none in the requirements document]*
`Deposit`	*[none in the requirements document]*
`BankDatabase`	authenticates a user, retrieves an account balance, credits a deposit amount to an account, debits a withdrawal amount from an account
`Account`	retrieves an account balance, credits a deposit amount to an account, debits a withdrawal amount from an account
`Screen`	displays a message to the user
`Keypad`	receives numeric input from the user
`CashDispenser`	dispenses cash, indicates whether it contains enough cash to satisfy a withdrawal request
`DepositSlot`	receives a deposit envelope

Fig. 12.16 | Verbs and verb phrases for each class in the ATM system.

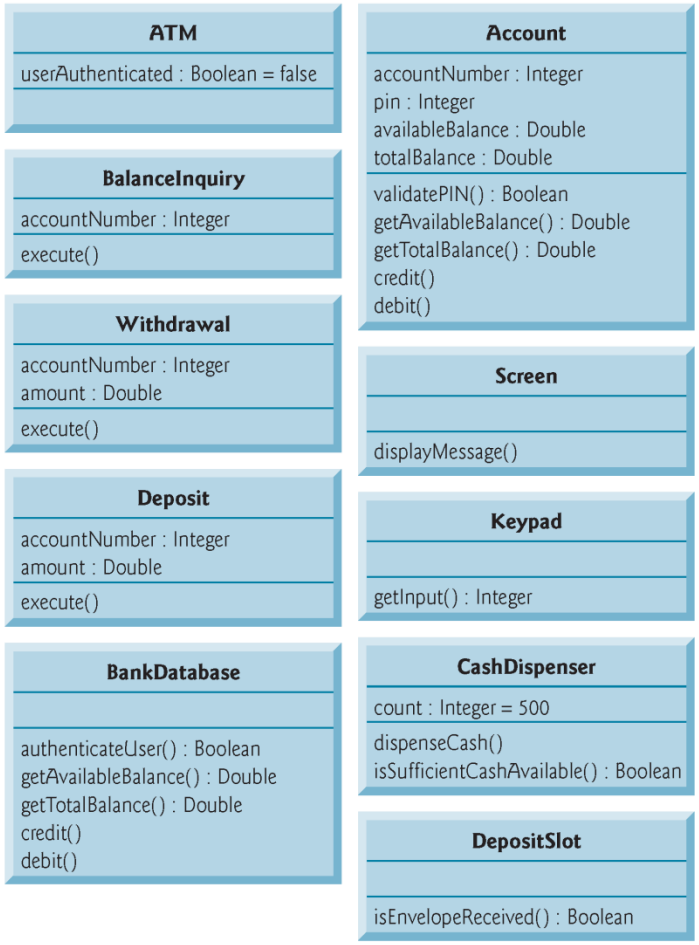


Fig. 12.17 | Classes in the ATM system with attributes and operations.

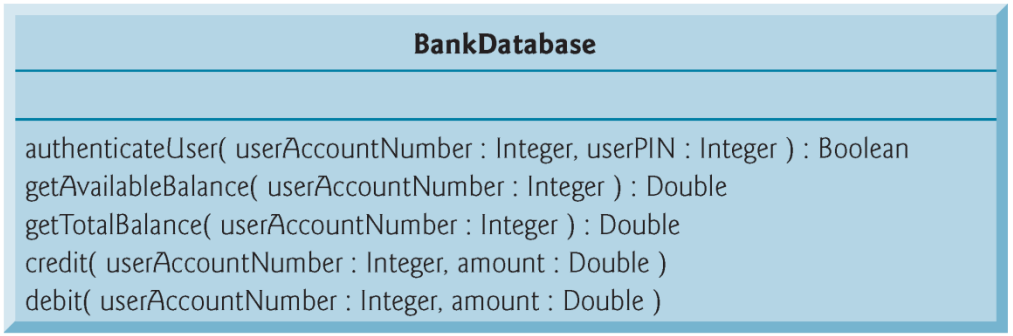


Fig. 12.18 | Class `BankDatabase` with operation parameters.

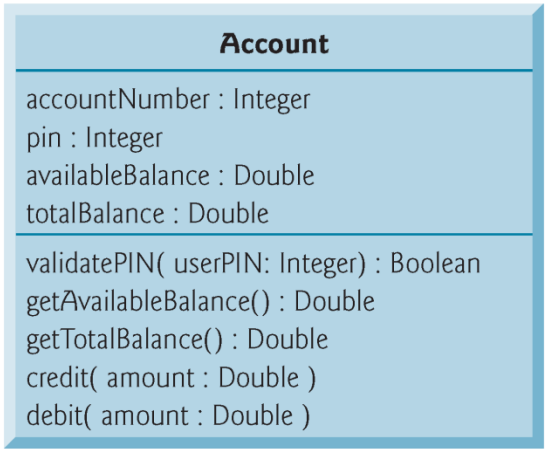


Fig. 12.19 | Class `Account` with operation parameters.

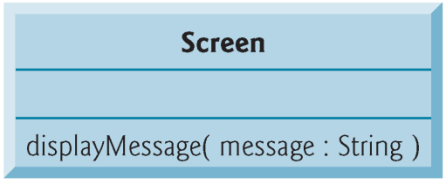


Fig. 12.20 | Class `Screen` with operation parameters.

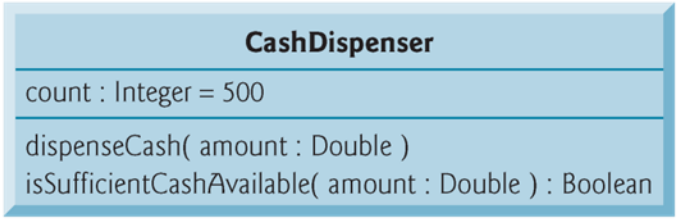


Fig. 12.21 | Class `CashDispenser` with operation parameters.

Identifying Collaborations in a system

- We identify the collaborations by reading the sections of the requirement document that specify what the ATM should do to perform each operation.
- For each action, we decide which objects must be involved.
- For objects involved, we identify the sending/receiving objects.
- Then we identify the operation of the receiving object invoked by the sending object.

Identifying Collaborations in a system

An object of class...	sends the message...	to an object of class...
ATM	displayMessage	Screen
	getInput	Keypad
	authenticateUser	BankDatabase
	execute	BalanceInquiry
	execute	Withdrawal
	execute	Deposit
BalanceInquiry	getAvailableBalance	BankDatabase
	getTotalBalance	BankDatabase
	displayMessage	Screen
Withdrawal	displayMessage	Screen
	getInput	Keypad
	getAvailableBalance	BankDatabase
	isSufficientCashAvailable	CashDispenser
	debit	BankDatabase
	dispenseCash	CashDispenser

Fig. 12.22 | Collaborations in the ATM system. (Part 1 of 2.)

Identifying Collaborations in a system

An object of class...	sends the message...	to an object of class...
Deposit	displayMessage	Screen
	getInput	Keypad
	isEnvelopeReceived	DepositSlot
	credit	BankDatabase
BankDatabase	validatePIN	Account
	getAvailableBalance	Account
	getTotalBalance	Account
	debit	Account
	credit	Account

Fig. 12.22 | Collaborations in the ATM system. (Part 2 of 2.)

Diagrams

- Communication diagrams
  - Model interactions among objects
  - Emphasis on ***what*** interactions occur.
- Sequence diagrams
  - Model interactions among objects
  - Emphasis on ***when*** interactions occur

5-Communication Diagram

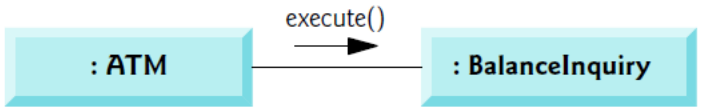


Fig. 12.23 | Communication diagram of the ATM executing a balance inquiry.

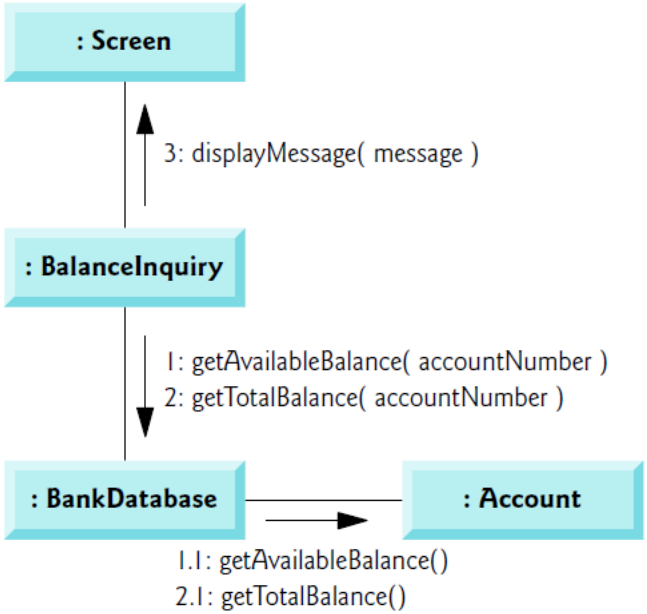


Fig. 12.24 | Communication diagram for executing a balance inquiry.

6-Sequence Diagram

Withdrawal Executing

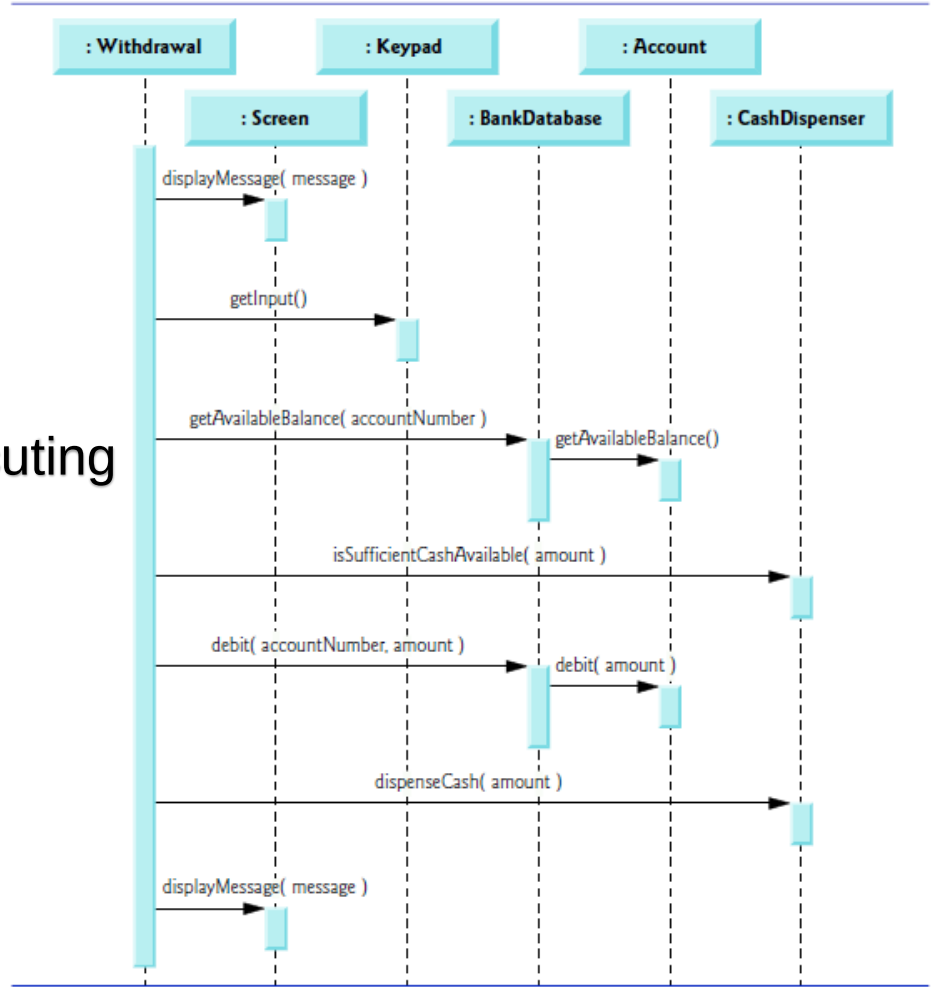


Fig. 12.25 | Sequence diagram that models a `Withdrawal` executing.

Deposit Executing

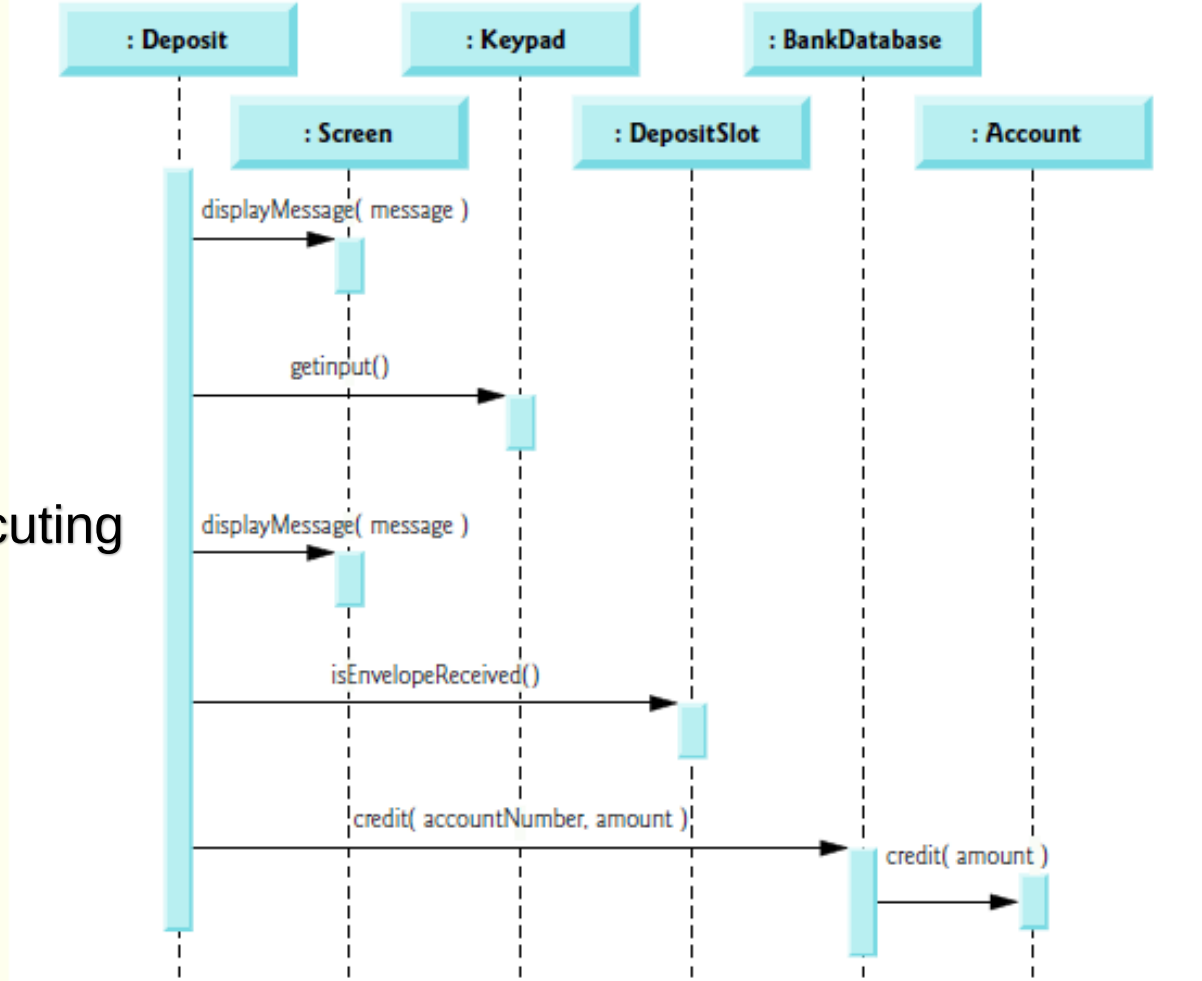
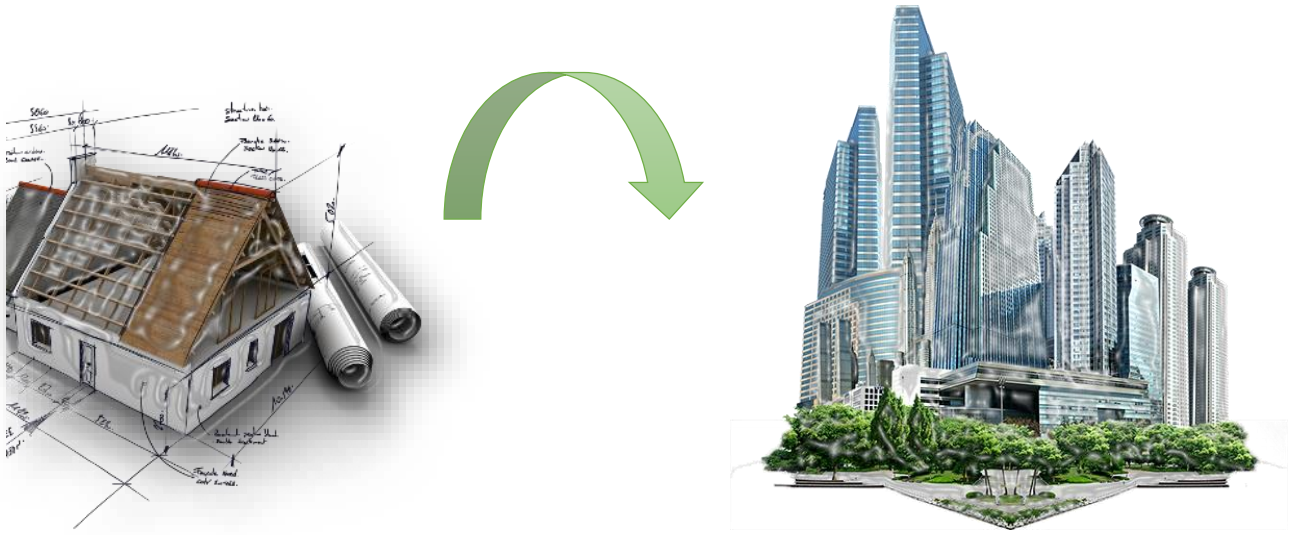


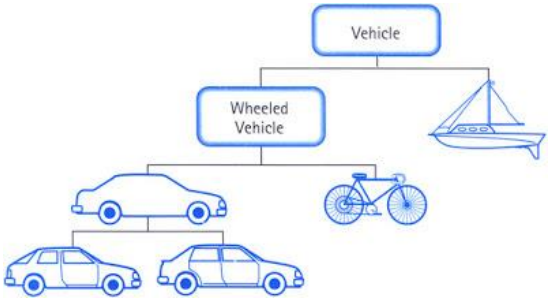
Fig. 12.30 | Sequence diagram that models a `Deposit` executing.

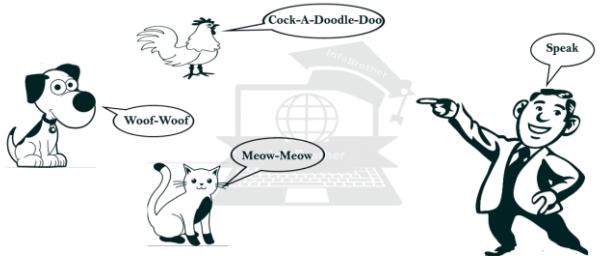
UML Diagrams to Implementation

- Convert class diagrams to Java code.
- Tune the design with inheritance and polymorphism.



Incorporating Inheritance and Polymorphism





- To apply inheritance, look for commonality among classes in the system.
- Create an inheritance hierarchy to model similar (yet not identical) classes in a more elegant and efficient manner.
- Modify class diagram to incorporate the new inheritance relationships.
- Polymorphism provides the `ATM` with an elegant way to execute all transactions “in the general.”
- The polymorphic approach also makes the system easily extensible.
- To create a new transaction type, just create an additional `Transaction` subclass that overrides the `execute` method with a version of the method appropriate for executing the new transaction type.

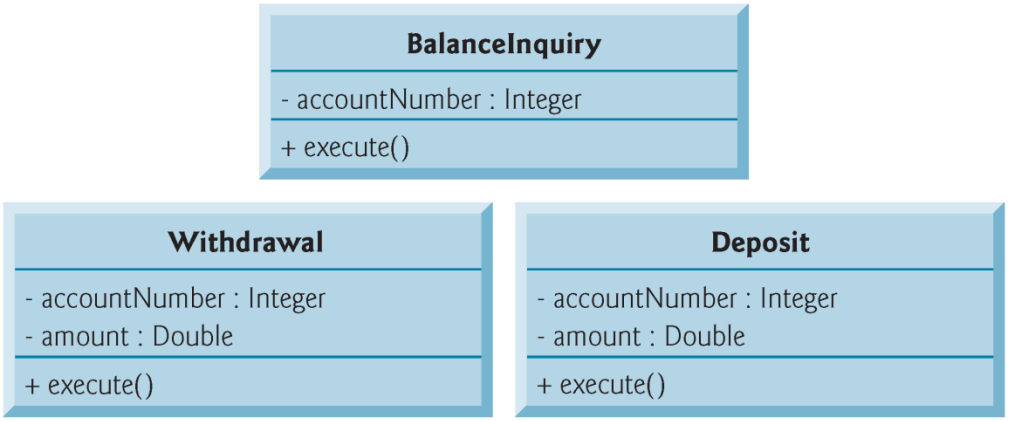


Fig. 13.7 | Attributes and operations of `BalanceInquiry`, `Withdrawal` and `Deposit`.

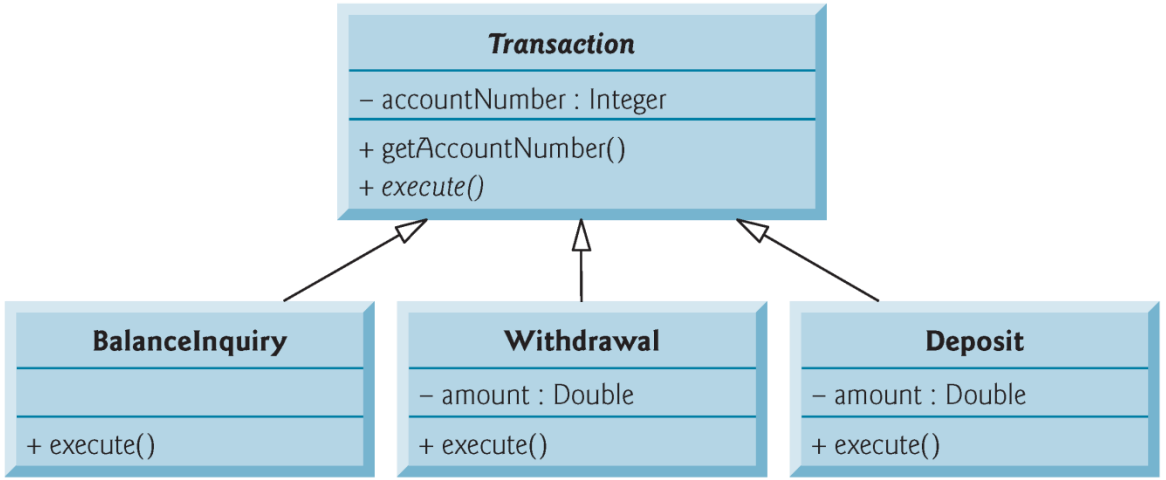


Fig. 13.8 | Class diagram modeling generalization of superclass `Transaction` and subclasses `BalanceInquiry`, `Withdrawal` and `Deposit`. Note that abstract

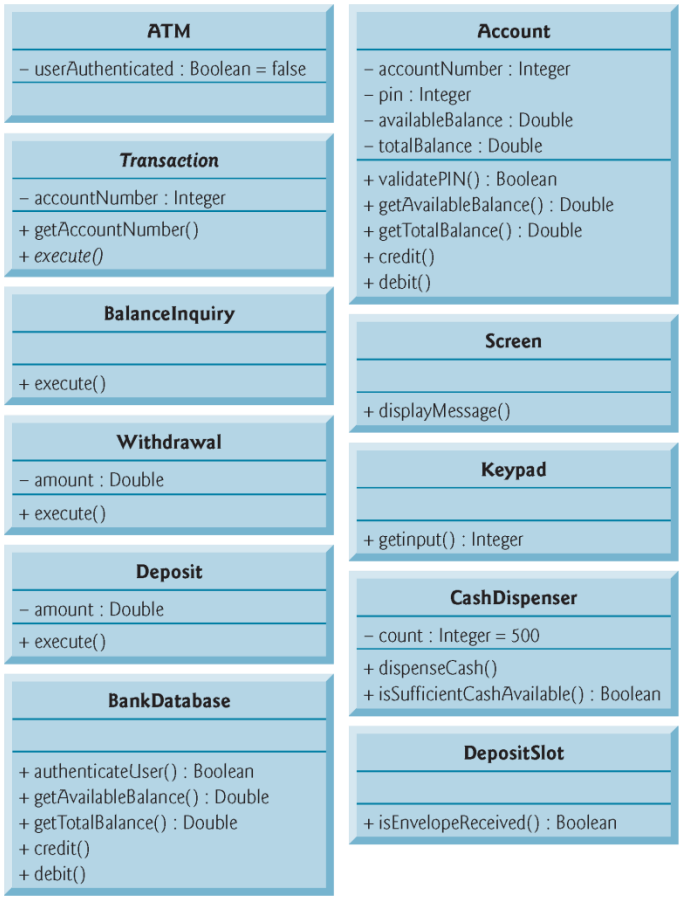


Fig. 13.10 | Class diagram with attributes and operations (incorporating inheritance). Note that the abstract class name `Transaction` and the abstract

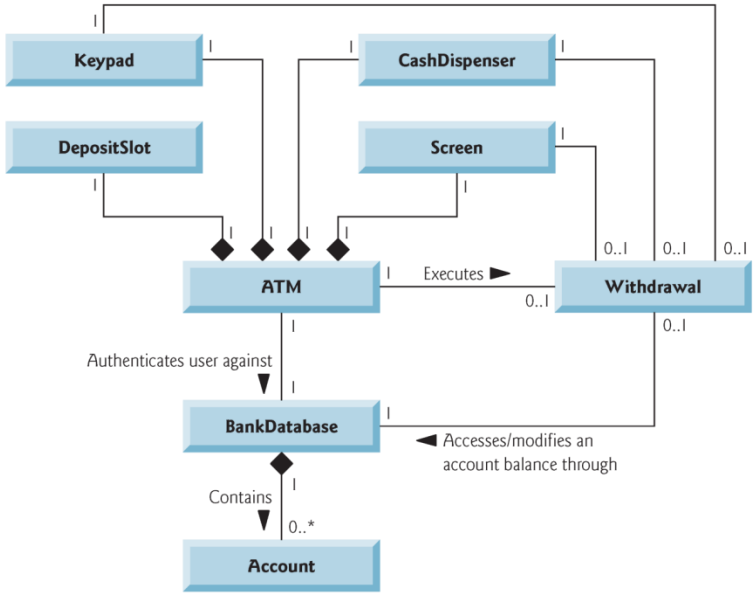


Fig. 12.10 | Class diagram for the ATM system model.

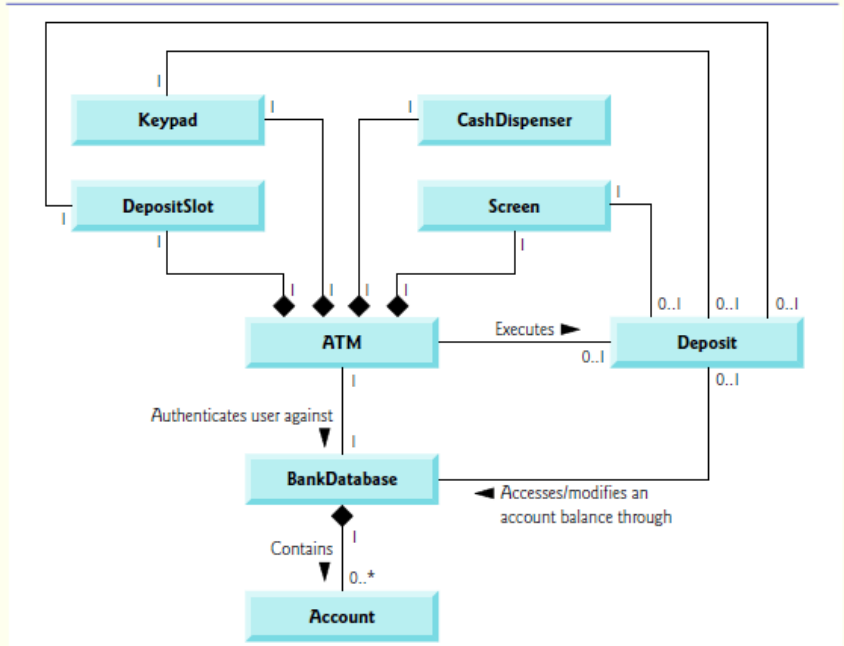


Fig. 12.28 | Class diagram for the ATM system model including class `Deposit`.

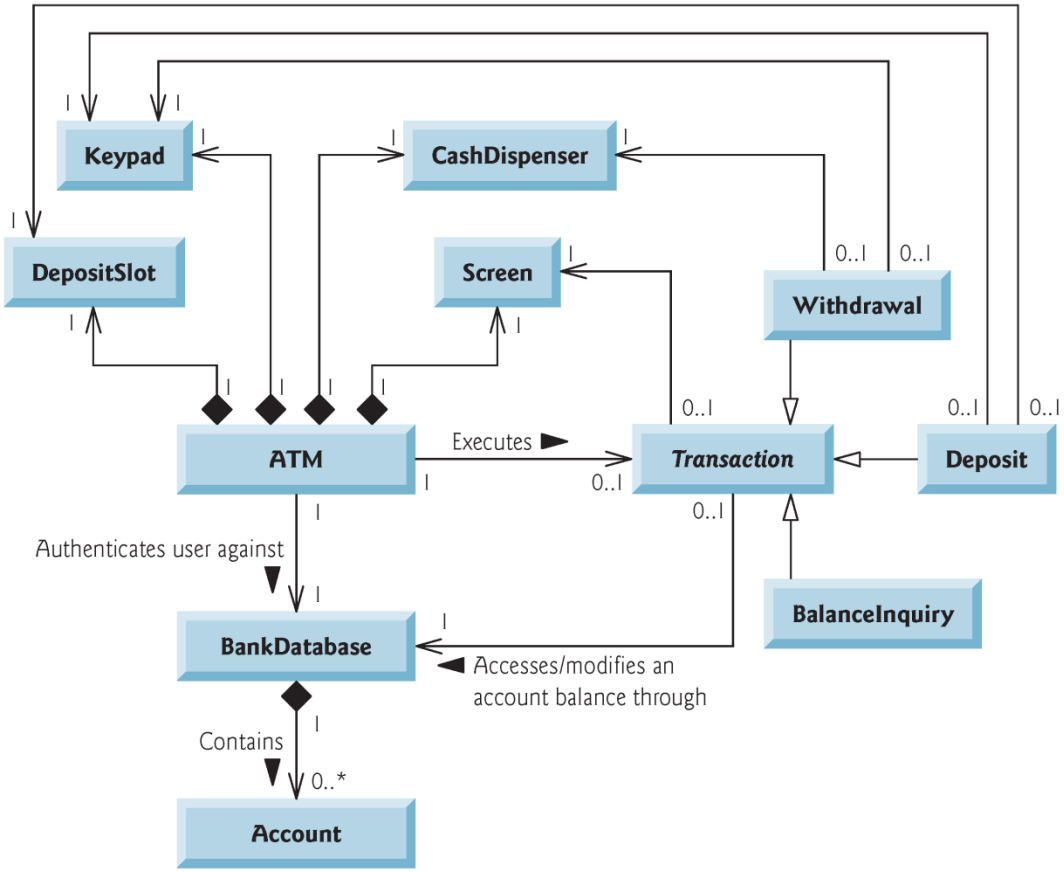


Fig. 13.9 | Class diagram of the ATM system (incorporating inheritance). Note that the abstract class name `Transaction` appears in italics.


THANK YOU!
TEAM
WORK
!?!