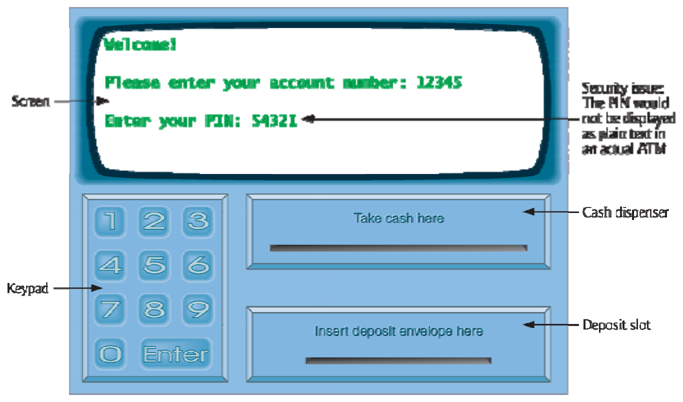
ATM Case Study Part 1

- A requirements document specifies the purpose of the ATM system and *what it must do.*

Requirements Document

- A local bank intends to install a new automated teller machine (ATM) to allow users (i.e., bank customers) to perform basic transactions
- Each user can have only one account at the bank.
- ATM users
  - view their account balance
  - withdraw cash
  - deposit funds

- ATM user interface:
  - a screen that displays messages to the user
  - a keypad that receives numeric input from the user
  - a cash dispenser that dispenses cash to the user and
  - a deposit slot that receives deposit envelopes from the user.
- The cash dispenser begins each day loaded with 500 $20 bills.


Welcome!

Please enter your account number: 12345

Enter your PIN: 54321

Screen
Keypad
Security issue: The PIN would not be displayed as plain text in an actual ATM
Cash dispenser
Deposit slot
Take cash here
Insert deposit envelope here
1
2
3
4
5
6
7
8
9
0
Enter

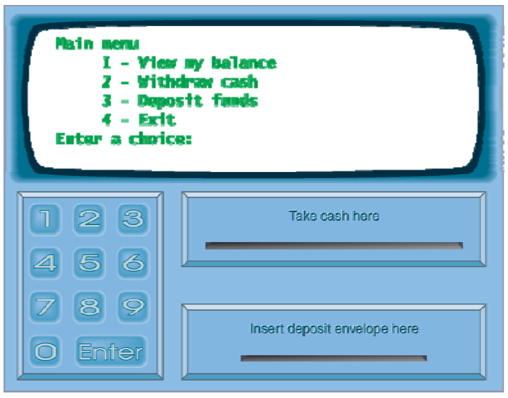
Fig. 12.1 | Automated teller machine user interface.

- Develop software to perform the financial transactions initiated by bank customers through the ATM.
  - The bank will integrate the software with the ATM's hardware at a later time.
- Software should encapsulate the functionality of the hardware devices within software components, but it need not concern itself with how these devices perform their duties.
- Use the computer's monitor to simulate the ATM's screen, and the computer's keyboard to simulate the ATM's keypad.

- An ATM session consists of
  - authenticating a user based on an account number and personal identification number (PIN)
  - creating and executing financial transactions
- To authenticate a user and perform transactions
  - interact with the bank's account information database
  - For each account, the database stores an account number, a PIN and a balance indicating the amount of money in the account.

- Simplifying Assumptions:
  - the bank plans to build only one ATM, so we need not worry about multiple ATMs accessing this database at the same time
  - the bank does not make any changes to the information in the database while a user is accessing the ATM.
  - an ATM faces reasonably complicated security issues that are beyond our scope.
  - the bank trusts the ATM to access and manipulate the information in the database without significant security measures.

- Upon first approaching the ATM, the user should experience the following sequence of events:
  - The screen displays `Welcome!` and prompts the user to enter an account number.
  - The user enters a five-digit account number using the keypad.
  - The screen prompts the user to enter the PIN (personal identification number) associated with the specified account number.
  - The user enters a five-digit PIN using the keypad.
  - If the user enters a valid account number and the correct PIN for that account, the screen displays the main menu. 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.



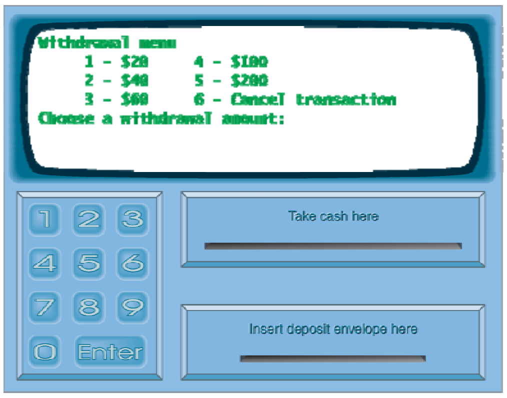
```
Main menu
      1 - View my balance
      2 - Withdraw cash
      3 - Deposit funds
      4 - Exit
Enter a choice:
```

Fig. 12.2 | ATM main menu.

- After the ATM authenticates the user, the main menu contains numbered options for the three types of transactions: balance inquiry (option 1), withdrawal (option 2) and deposit (option 3).
- It also should contain an option to allow the user to exit the system (option 4).
- The user then chooses either to perform a transaction (by entering 1, 2 or 3) or to exit the system (by entering 4).
- If the user enters 1 to make a balance inquiry, the screen displays the user's account balance.
- To do so, the ATM must retrieve the balance from the bank's database.

- The following steps describe what occurs when the user enters 2 to make a withdrawal:
  - The screen displays a menu of standard withdrawal amounts and an option to cancel the transaction.
  - The user enters a menu selection using the keypad.
  - If the withdrawal amount is greater than the user's account balance, the screen displays a message stating this and telling the user to choose a smaller amount. The ATM then returns to *Step 1*. If the withdrawal amount chosen is less than or equal to the user's account balance (i.e., an acceptable amount), the ATM proceeds to *Step 4*. If the user chooses to cancel, the ATM displays the main menu and waits for user input.

  - If the cash dispenser contains enough cash, the ATM proceeds to *Step 5*. Otherwise, the screen displays a message indicating the problem and telling the user to choose a smaller withdrawal amount. The ATM then returns to *Step 1*.
  - 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 to the user.
  - The screen displays a message reminding the user to take the money.



```
Withdrawal menu
      1 - $20      4 - $100
      2 - $40      5 - $200
      3 - $60      6 - Cancel transaction
Choose a withdrawal amount:
```

Fig. 12.3 | ATM withdrawal menu.

- The following steps describe the actions that occur 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 to *Step 4*. If the user chooses to cancel the transaction (by entering 0), the ATM displays the main menu and waits for user input.
  - 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 (i.e., adds the deposit amount to the user's account balance).

- After the system successfully executes a transaction, it should return to the main menu so that the user can perform additional transactions.
- If the user exits the system, the screen should display a thank you message, then display the welcome message for the next user.

Analyzing the ATM System

- The preceding statement is a simplified example of a requirements document
  - Typically the result of a detailed process of requirements gathering
- A systems analyst
  - might interview banking experts to gain a better understanding of what the software must do
  - would use the information gained to compile a list of system requirements to guide systems designers as they design the system.

- The software life cycle specifies the stages through which software goes from the time it’s first conceived to the time it’s retired from use.
  - These stages typically include: analysis, design, implementation, testing and debugging, deployment, maintenance and retirement.
- Several software life-cycle models exist
  - Waterfall models perform each stage once in succession
  - Iterative models may repeat one or more stages several times throughout a product’s life cycle

- The analysis stage focuses on defining the problem to be solved.
- When designing any system, one must *solve the problem right*, but of equal importance, *one must solve the right problem.*
- Our requirements document describes the requirements of our ATM system in sufficient detail that you need not go through an extensive analysis stage—it’s been done for you.

- Use case modeling identifies the use cases of the system, each representing a different capability that the system provides to its clients.
  - “View Account Balance”
  - “Withdraw Cash”
  - “Deposit Funds”
- Each use case describes a typical scenario for which the user uses the system.

- A use case diagram models the interactions between a system’s clients and its use cases.
- Shows the kinds of interactions users have with a system without providing the details
- Often accompanied by informal text that gives more detail—like the text that appears in the requirements document.
- Produced during the analysis stage of the software life cycle.
- Stick figure represents an actor, which defines the roles that an external entity—such as a person or another system—plays when interacting with the system.

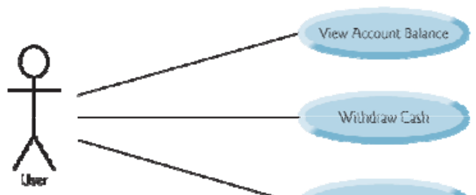


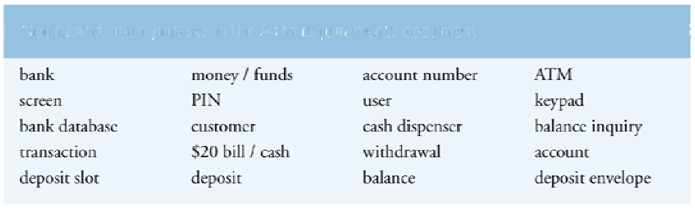
Fig. 12.4 | Use case diagram for the ATM system from the User's perspective.

- During the analysis stage, systems designers focus on understanding the requirements document to produce a high-level specification that describes *what* the system is supposed to do.
- The output of the design stage—a design specification - specifies clearly *how* the system should be constructed to satisfy these requirements.
- In the next several sections, we perform the steps of a simple object-oriented design (OOD) process on the ATM system to produce a design specification containing a collection of UML diagrams and supporting text.
- We present our own simplified design process

- A system is a set of components that interact to solve a problem.
- System structure describes the system's objects and their interrelationships.
- System behavior describes how the system changes as its objects interact with one another.
- Every system has both structure and behavior—designers must specify both.

- The UML 2 standard specifies 13 diagram types for documenting the system models.
- Each models a distinct characteristic of a system's structure or behavior—six diagrams relate to system structure, the remaining seven to system behavior.
- We are interested in two of the six diagram types:
  - Use case diagrams model the interactions between a system and its external entities (actors) in terms of use cases.
  - Class diagrams model the classes, or "building blocks," used in a system.

- Identify the classes that are needed to build the system by analyzing the *nouns and noun phrases* that appear in the requirements document.
- We introduce UML class diagrams to model these classes.
  - Important first step in defining the system’s structure.
- Review the requirements document and identify key nouns and noun phrases to help us identify classes that comprise the ATM system.
  - We may decide that some of these nouns and noun phrases are actually attributes of other classes in the system.
  - We may also conclude that some of the nouns do not correspond to parts of the system and thus should not be modeled at all.
  - 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.

- We create classes only for the nouns and noun phrases that have significance in the ATM system.
- Though the requirements document frequently describes a "transaction" in a general sense, we do not model the broad notion of a financial transaction at this time.
  - Instead, we model the three types of transactions (i.e., "balance inquiry," "withdrawal" and "deposit") as individual classes.
  - These classes possess specific attributes needed for executing the transactions they represent.

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

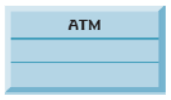


Fig. 12.6 | Representing a class in the UML using a class diagram.

- The UML enables us to model, via class diagrams, the classes in the ATM system and their interrelationships.
- Each class is modeled as a rectangle with three compartments.
  - The top one contains the name of the class centered horizontally in boldface.
  - The middle compartment contains the class's attributes.
  - The bottom compartment contains the class's operations.

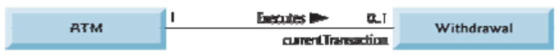


Fig. 12.7 | Class diagram showing an association among classes.

- The UML allows the suppression of class attributes and operations in this manner to create more readable diagrams, when appropriate.
- Class diagrams also show the relationships between the classes of the system. The solid line that connects the two classes represents an association between classes.
- The numbers near each end of the line are multiplicity values, which indicate how many objects of each class participate in the association.

- An association can be named.
  - The word `Executes` above the line connecting classes `ATM` and `Withdrawal` in Fig. 12.7 indicates the name of that association.
  - This part of the diagram reads “one object of class `ATM` executes zero or one objects of class `Withdrawal`.”
- Association names are *directional*, as indicated by the filled arrowhead.
- The word `currentTransaction` at the `Withdrawal` end of the association line is a role name, identifying the role the `Withdrawal` object plays in its relationship with the `ATM`.
- A role name adds meaning to an association between classes by identifying the role a class plays in the context of an association.
- A class can play several roles in the same system.
- Role names in class diagrams are often omitted when the meaning of an association is clear without them.

- Solid diamonds attached to the ATM class’s association lines indicate that ATM has a composition relationship with classes `Screen`, `Keypad`, `CashDispenser` and `DepositSlot`.
- Composition implies a whole/part relationship.
- The class that has the composition symbol (the solid diamond) on its end of the association line is the whole (in this case, ATM), and the classes on the other end of the association lines are the parts.

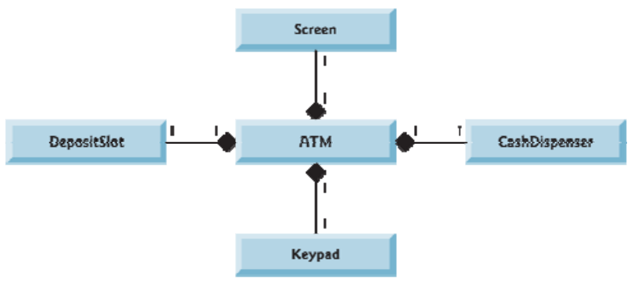


Fig. 12.9 | Class diagram showing composition relationships.

- Composition relationships have the following properties:
  - Only one class in the relationship can represent the whole
  - The parts in the composition relationship exist only as long as the whole does, and the whole is responsible for the creation and destruction of its parts.
  - A part may belong to only one whole at a time, although it may be removed and attached to another whole, which then assumes responsibility for the part.
- If a *has-a* relationship does not satisfy one or more of these criteria, the UML specifies that hollow diamonds be attached to the ends of association lines to indicate aggregation—a weaker form of composition.

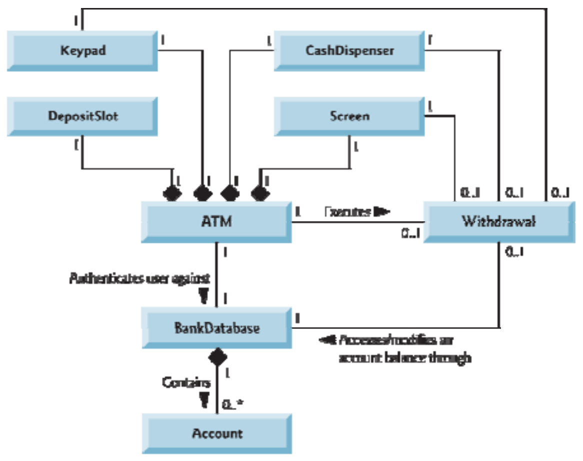
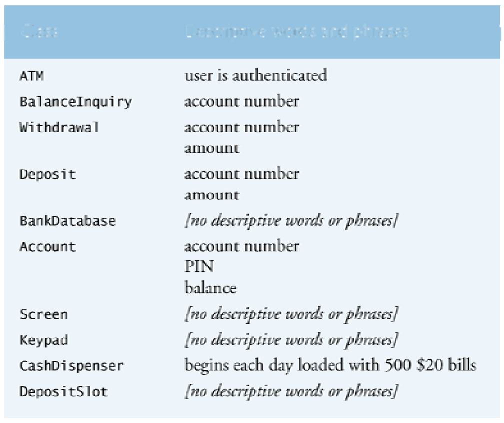


Fig. 12.10 | Class diagram for the ATM system model.

- Class `ATM` has a one-to-one relationship with class `BankDatabase`—one `ATM` object authenticates users against one `BankDatabase` object.
- The bank's database contains information about many accounts—one `BankDatabase` object participates in a composition relationship with zero or more `Account` objects.
  - The multiplicity value 0..* at the `Account` end of the association between class `BankDatabase` and class `Account` indicates that zero or more objects of class `Account` take part in the association.
- Class `BankDatabase` has a one-to-many relationship with class `Account`—the `BankDatabase` contains many `Account`s.
- Class `Account` has a many-to-one relationship with class `BankDatabase`—there can be many `Account`s stored in the `BankDatabase`.

- At any given time 0 or 1 `Withdrawal` objects can exist.
- If the user is performing a withdrawal, “one object of class `Withdrawal` accesses/modifies an account balance through one object of class `BankDatabase`.”
- All other parts of the system must interact with the database to retrieve or update account information.

- Classes have attributes (data) and operations (behaviors).
- Class attributes are implemented in Java programs as fields, and class operations are implemented as methods.
- In this section, we determine many of the attributes needed in the ATM system.
- 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 identified earlier.
- We also create attributes to represent any additional data that a class may need, as such needs become clear.
- Next the list the words or phrases from the requirements document that describe each class.



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.

- For real problems in industry, there is no guarantee that requirements documents will be precise enough for the object-oriented systems designer to determine all the attributes or even all the classes.
- The need for additional classes, attributes and behaviors may become clear as the design process proceeds.

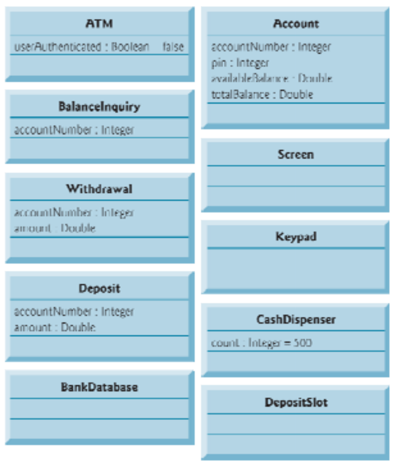


Fig. 12.12 | **Classes with attributes.**

- Attributes represent an object’s state.
- We identify some key states that our objects may occupy and discuss how objects change state in response to various events occurring in the system.
- We also discuss the workflow, or activities, that objects perform in the ATM system, and we present the activities of `BalanceInquiry` and `Withdrawal` transaction objects.

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.**

- 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.

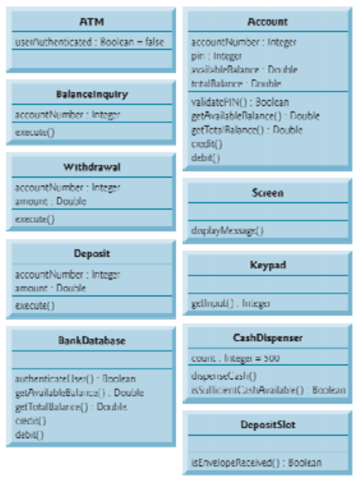


Fig. 12.17 | **Classes in the ATM system with attributes and operations.**

- The UML represents operations (methods in Java) by listing the operation name, followed by a comma-separated list of parameters in parentheses, a colon and the return type:
  - *operationName(parameter1, parameter2, …, parameterN) : return type*
- Each parameter in the comma-separated parameter list consists of a parameter name, followed by a colon and the parameter type:
  - *parameterName : parameterType*

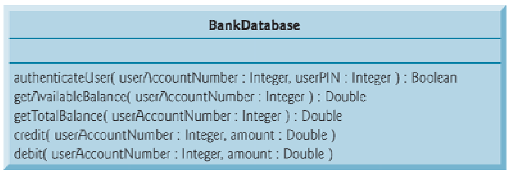

BankDatabase
authenticateUser(userAccountNumber : Integer, userPIN : Integer) : Boolean
getAvailableBalance(userAccountNumber : Integer) : Double
getTotalBalance(userAccountNumber : Integer) : Double
credit(userAccountNumber : Integer, amount : Double)
debit(userAccountNumber : Integer, amount : Double)

Fig. 12.18 | Class BankDatabase with operation parameters.

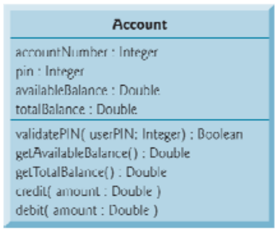

Account
accountNumber : Integer
pin : Integer
availableBalance : Double
totalBalance : Double
validatePIN(userPIN: Integer) : Boolean
getAvailableBalance() : Double
getTotalBalance() : Double
credit(amount : Double)
debit(amount : Double)

Fig. 12.19 | Class Account with operation parameters.

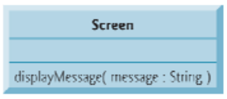


Fig. 12.20 | Class Screen with operation parameters.

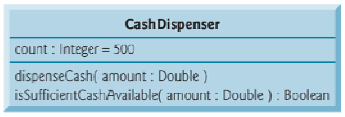

CashDispenser
count : Integer = 500
dispenseCash(amount : Double)
isSufficientCashAvailable(amount : Double) : Boolean

Fig. 12.21 | Class CashDispenser with operation parameters.

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.)

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.)

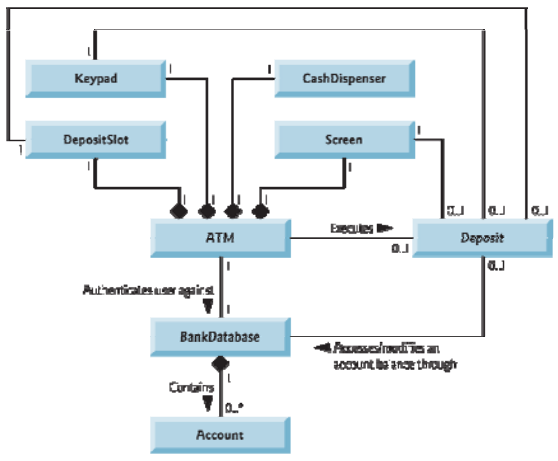


Fig. 12.28 | Class diagram for the ATM system model including class Deposit.