Software Engineering

Object-Oriented Analysis, Design and Implementation

Case Study
Part I

Assoc. Prof. Dr. Marenglen Biba
MSc in Computer Science, UoG-UNYT
Foundation Programme

ATM Case Study, Part 1: Object-Oriented Design with the UML

Assoc. Prof. Marenglen Biba

(C) 2010 Pearson Education, Inc. All rights reserved.

In this chapter you'll learn:

- A simple object-oriented design methodology.
- What a requirements document is.
- To identify classes and class attributes from a requirements document.
- To identify objects' states, activities and operations from a requirements document.
- To determine the collaborations among objects in a system.
- To work with the UML's use case, class, state, activity, communication and sequence diagrams to graphically model an object-oriented system.

(C) 2010 Pearson Education, Inc. All rights reserved.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.1 Case Study Introduction

- In Chapters 12–13, you design and implement an object-oriented automated teller machine (ATM) software system.
- Concise, carefully paced, complete design and implementation experience.
- Perform the steps of an object-oriented design (OOD) process using the UML
  - Sections 12.2—12.7 and 13.2–13.3
  - Relate these steps to the object-oriented concepts discussed in Chapters 2–10
- Work with UML diagrams
- Chapter 13—tune the design with inheritance, then fully implement the ATM.
- An end-to-end learning experience that concludes with a detailed walkthrough of the complete Java code.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.2 Examining the Requirements Document

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

(C) 2010 Pearson Education, Inc. All rights reserved.

12.2 Examining the Requirements Document (cont.)

- 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.

(C) 2010 Pearson Education, Inc. All rights reserved.

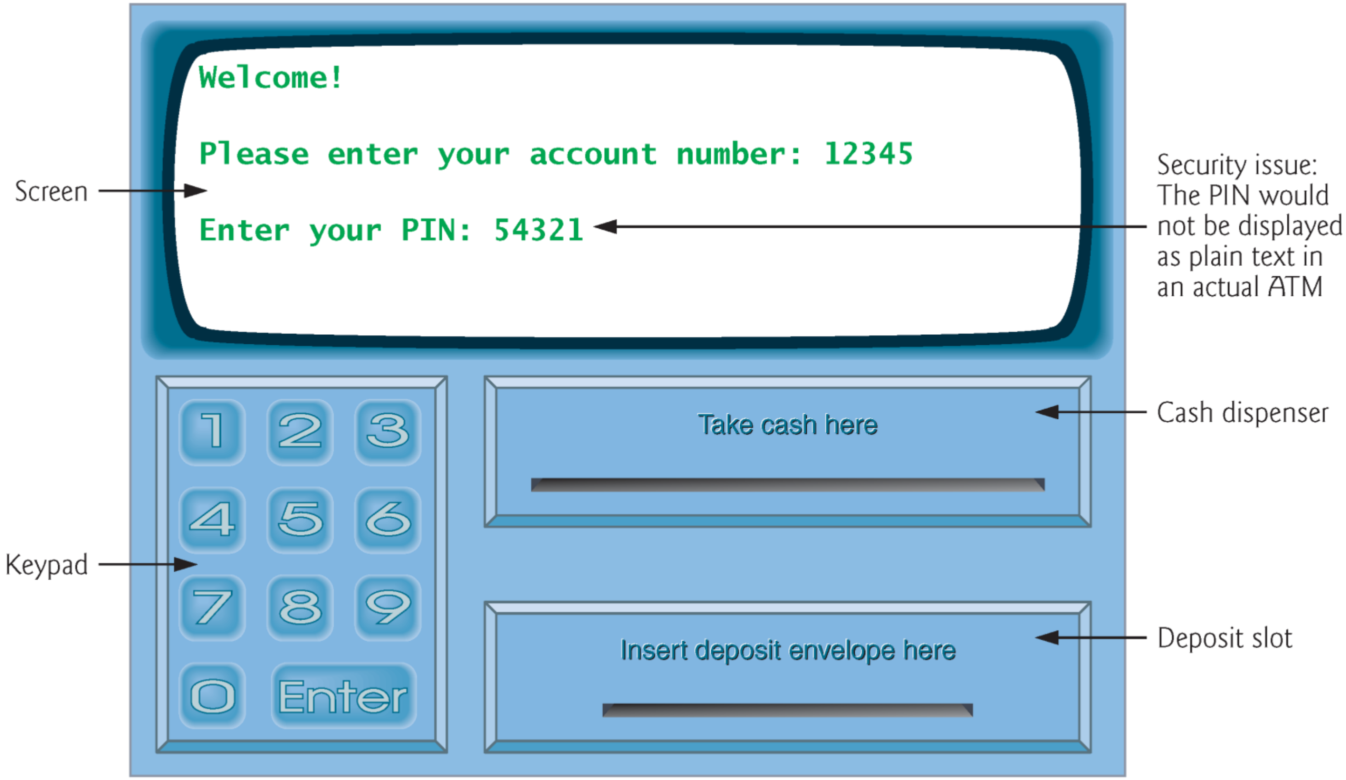

Fig. 12.1 | Automated teller machine user interface.

(C) 2010 Pearson Education, Inc. All rights reserved.

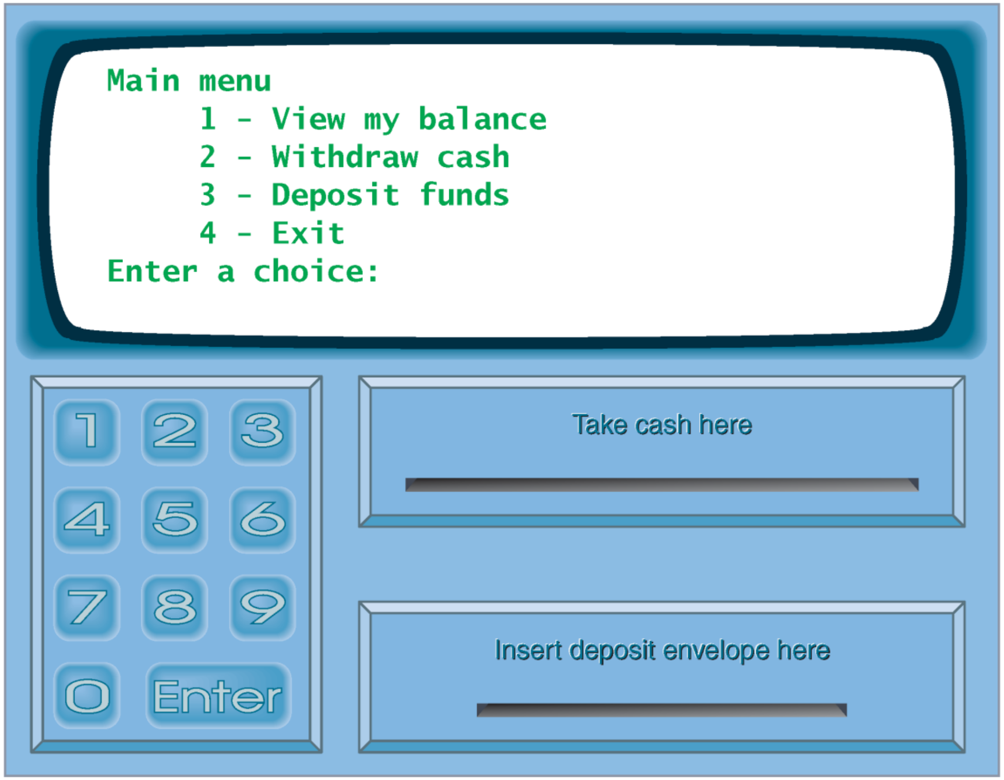

Fig. 12.2 | ATM main menu.

(C) 2010 Pearson Education, Inc. All rights reserved.

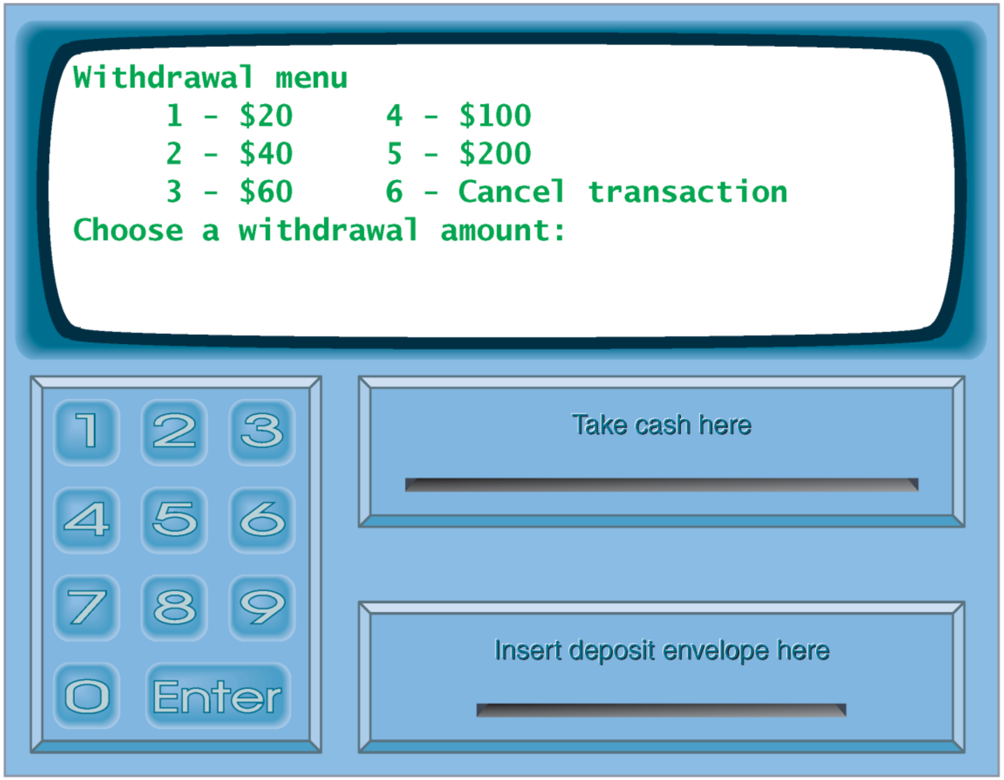

Fig. 12.3 | ATM withdrawal menu.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.2 Examining the Requirements Document (cont.)

- 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.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.2 Examining the Requirements Document (cont.)

- 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.

(C) 2010 Pearson Education, Inc. All rights reserved.

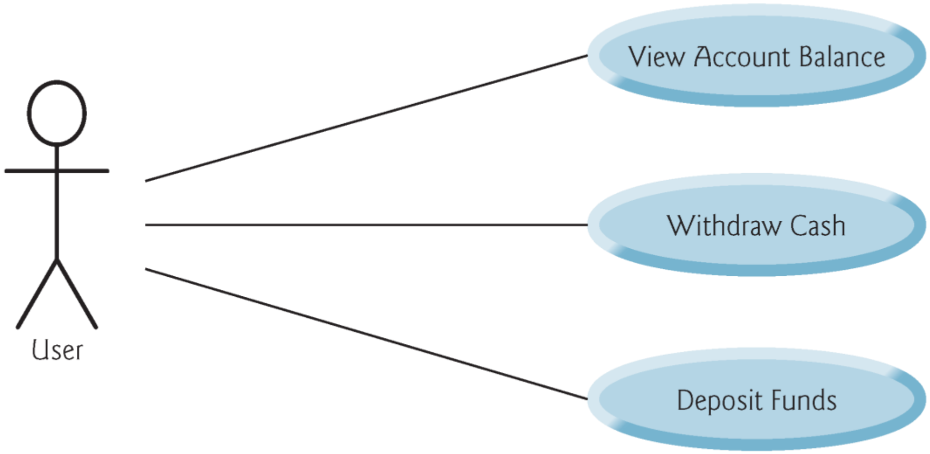

Fig. 12.4 | Use case diagram for the ATM system from the User's perspective.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.2 Examining the Requirements Document (cont.)

- 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 list only the two diagram types used in our case study.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.2 Examining the Requirements Document (cont.)

- 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.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.3 Identifying the Classes in a Requirements Document

- 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.

(C) 2010 Pearson Education, Inc. All rights reserved.

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.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.3 Identifying the Classes in a Requirements Document (cont.)

- 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.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.3 Identifying the Classes in a Requirements Document (cont.)

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

(C) 2010 Pearson Education, Inc. All rights reserved.

12.3 Identifying the Classes in a Requirements Document (cont.)

- The UML enables us to model, via class diagrams, the classes in the ATM system and their interrelationships.
- Figure 12.6 represents class ATM.
- 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.

(C) 2010 Pearson Education, Inc. All rights reserved.

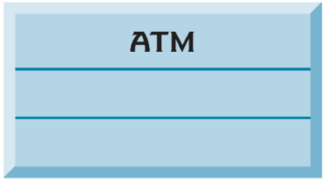

Fig. 12.6 | Representing a class in the UML using a class diagram.

(C) 2010 Pearson Education, Inc. All rights reserved.

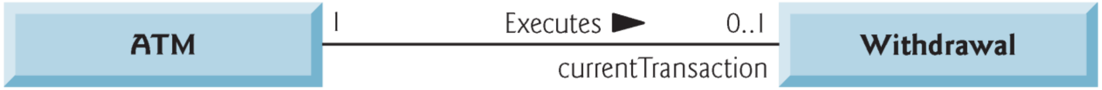

Fig. 12.7 | Class diagram showing an association among classes.

(C) 2010 Pearson Education, Inc. All
rights reserved.

12.3 Identifying the Classes in a Requirements Document (cont.)

- The solid line that connects the two classes represents an association—a relationship 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.
  - At any given moment, one `ATM` object participates in an association with either zero or one `Withdrawal` objects—zero if the current user is not currently performing a transaction or has requested a different type of transaction, and one if the user has requested a withdrawal.
- Figure 12.8 lists and explains the multiplicity types.

(C) 2010 Pearson Education, Inc. All rights reserved.

Symbol	Meaning
0	None
1	One
m	An integer value
0..1	Zero or one
m, *n*	*m* or *n*
m..*n*	At least *m*, but not more than *n*
*	Any nonnegative integer (zero or more)
0..*	Zero or more (identical to *)
1..*	One or more

Fig. 12.8 | Multiplicity types.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.3 Identifying the Classes in a Requirements Document (cont.)

- In Fig. 12.9, the 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.

(C) 2010 Pearson Education, Inc. All rights reserved.

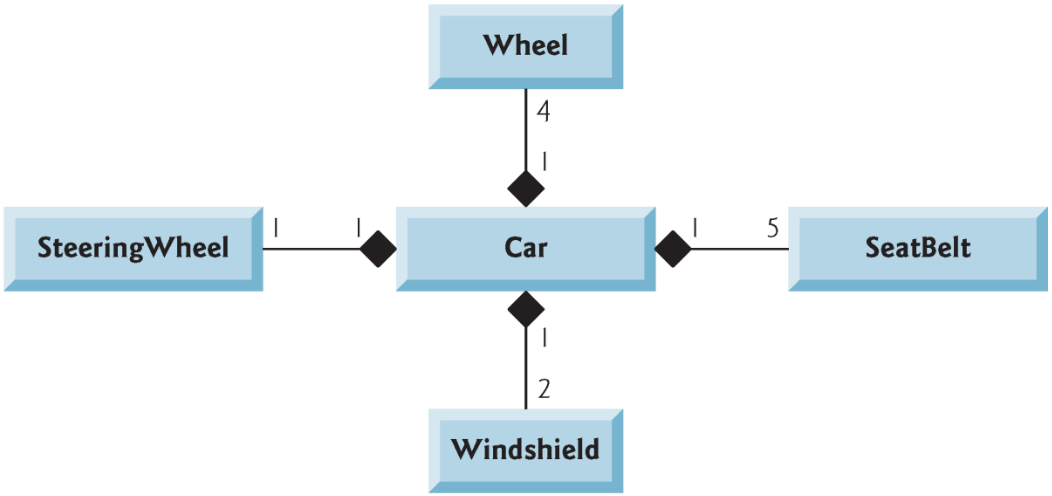

Fig. 12.27 | Class diagram showing composition relationships of a class `Car`.

(C) 2010 Pearson Education, Inc. All rights reserved.

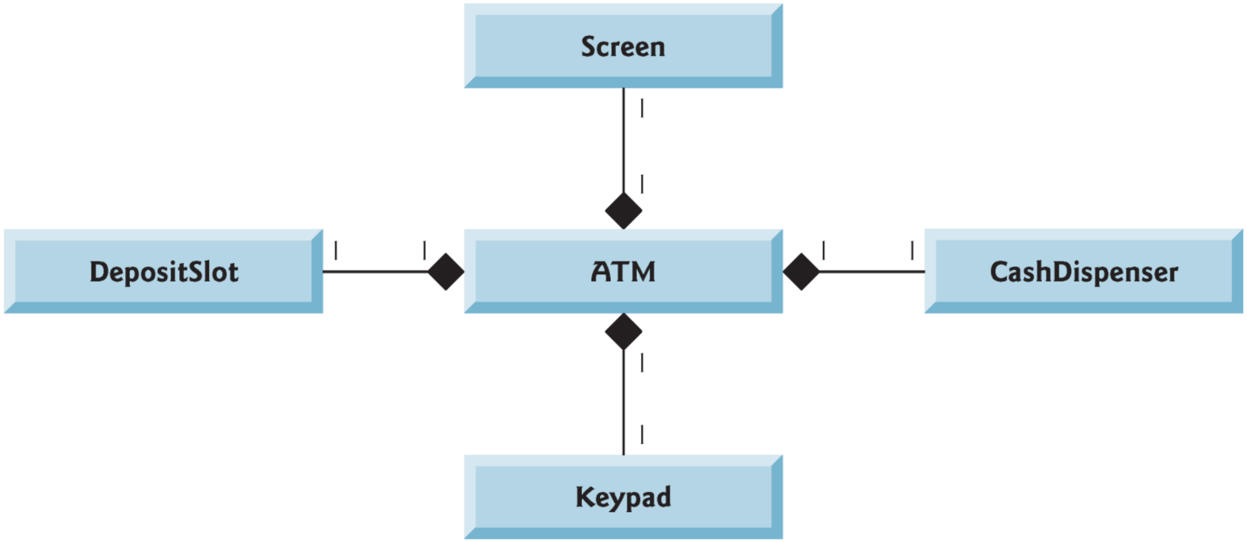

Fig. 12.9 | Class diagram showing composition relationships.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.3 Identifying the Classes in a Requirements Document (cont.)

- 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.

(C) 2010 Pearson Education, Inc. All rights reserved.

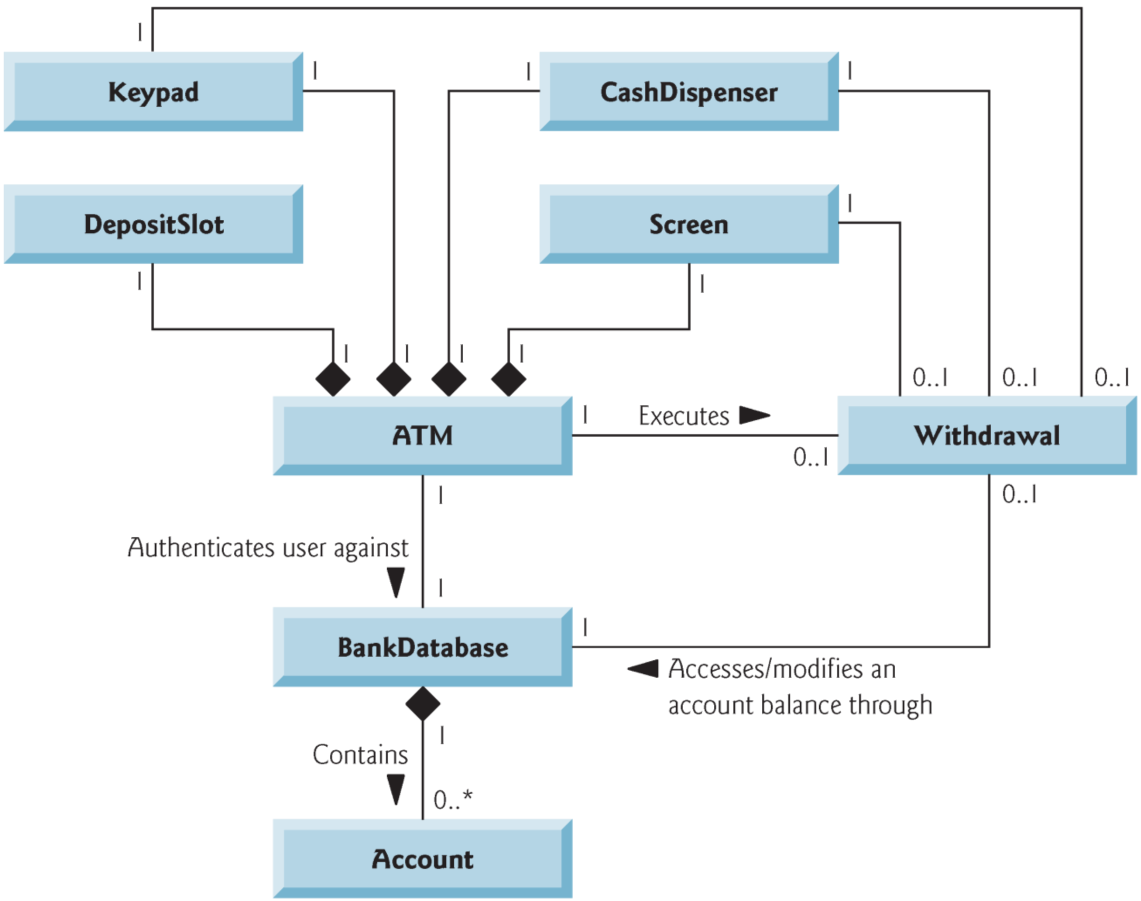

Fig. 12.10 | Class diagram for the ATM system model.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.4 Identifying Class Attributes

- 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.

(C) 2010 Pearson Education, Inc. All rights reserved.

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.

(C) 2010 Pearson Education, Inc. All rights reserved.

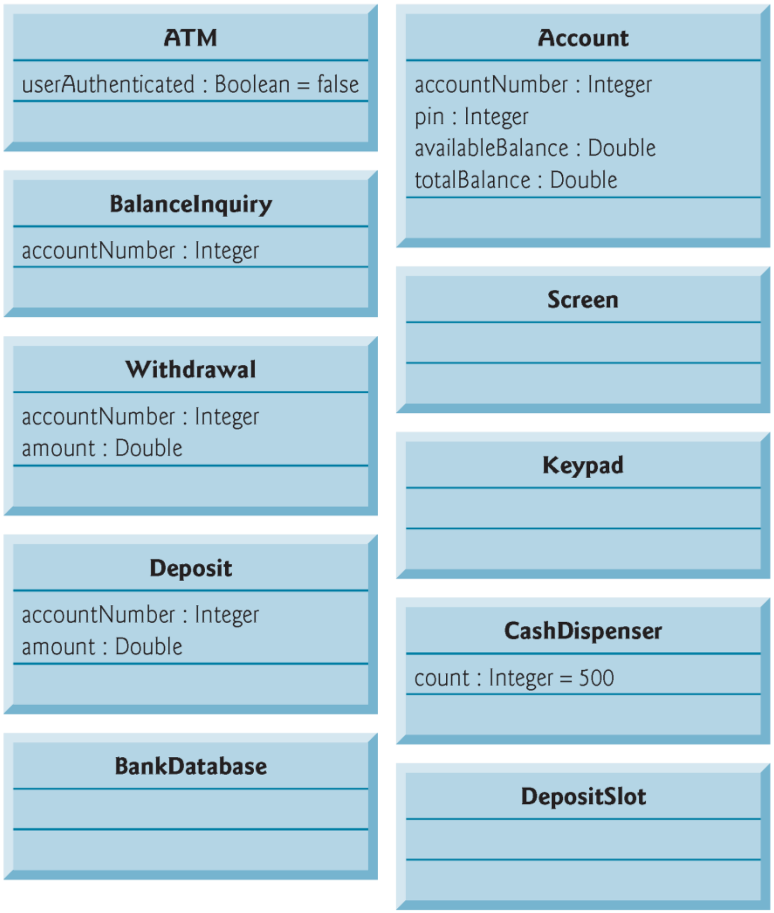

Fig. 12.12 | Classes with attributes.

(C) 2010 Pearson Education, Inc. All rights reserved.

12.6 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.
- The verb phrases in Fig. 12.16 help us determine the operations of each class.

(C) 2010 Pearson Education, Inc. All rights reserved.

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.

(C) 2010 Pearson Education, Inc. All rights reserved.

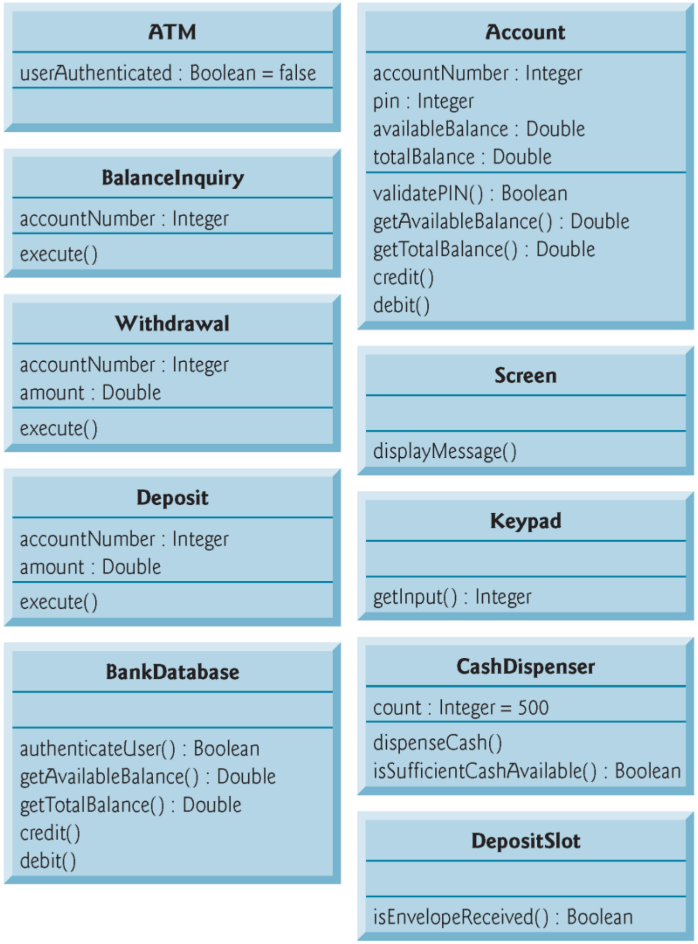

Fig. 12.17 | Classes in the ATM system with attributes and operations.

(C) 2010 Pearson Education, Inc. All rights reserved.

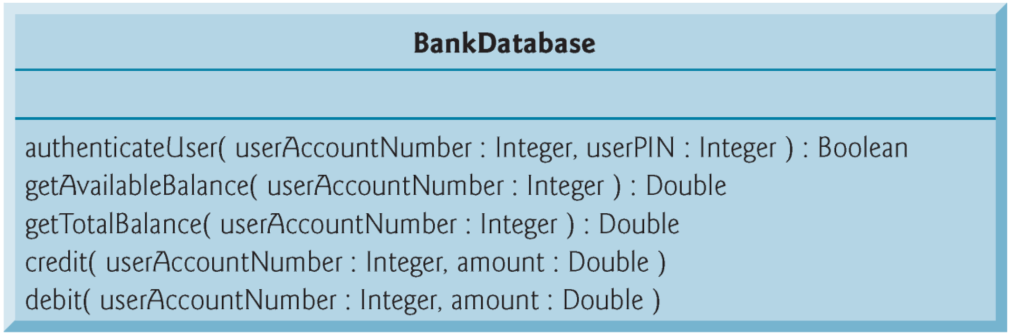

Fig. 12.18 | Class `BankDatabase` with operation parameters.

(C) 2010 Pearson Education, Inc. All rights reserved.

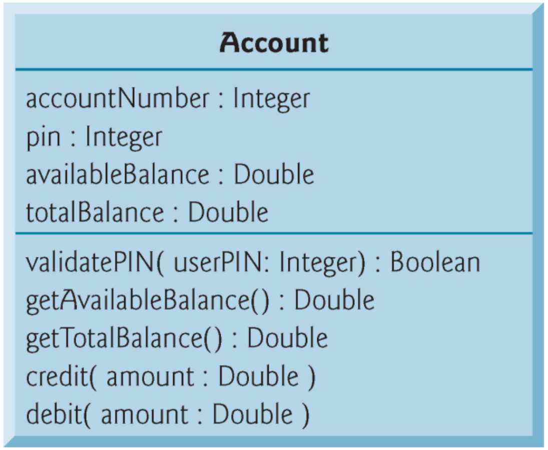

Fig. 12.19 | Class `Account` with operation parameters.

(C) 2010 Pearson Education, Inc. All rights reserved.

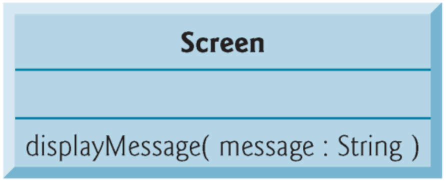

Fig. 12.20 | Class `Screen` with operation parameters.

(C) 2010 Pearson Education, Inc. All rights reserved.

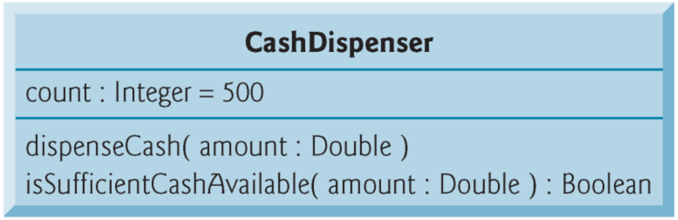

Fig. 12.21 | Class `CashDispenser` with operation parameters.

(C) 2010 Pearson Education, Inc. All rights reserved.

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.)

(C) 2010 Pearson Education, Inc. All rights reserved.

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.)

(C) 2010 Pearson Education, Inc. All rights reserved.

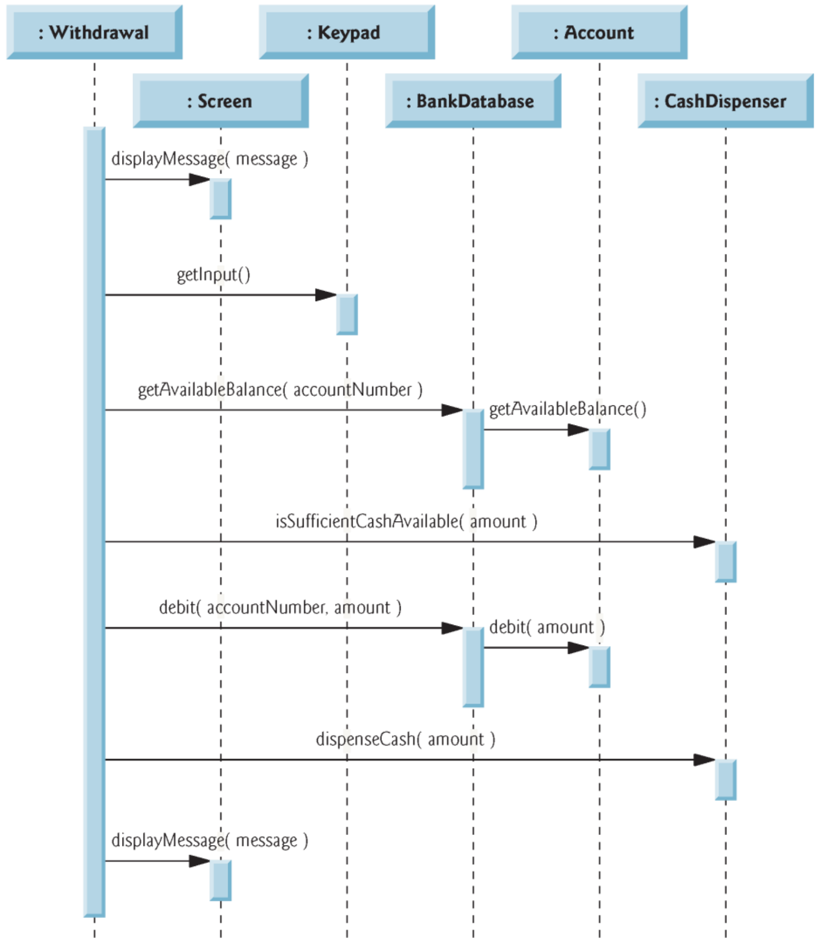

Fig. 12.25 | Sequence diagram that models a `Withdrawal` executing.

(C) 2010 Pearson Education, Inc. All rights reserved.

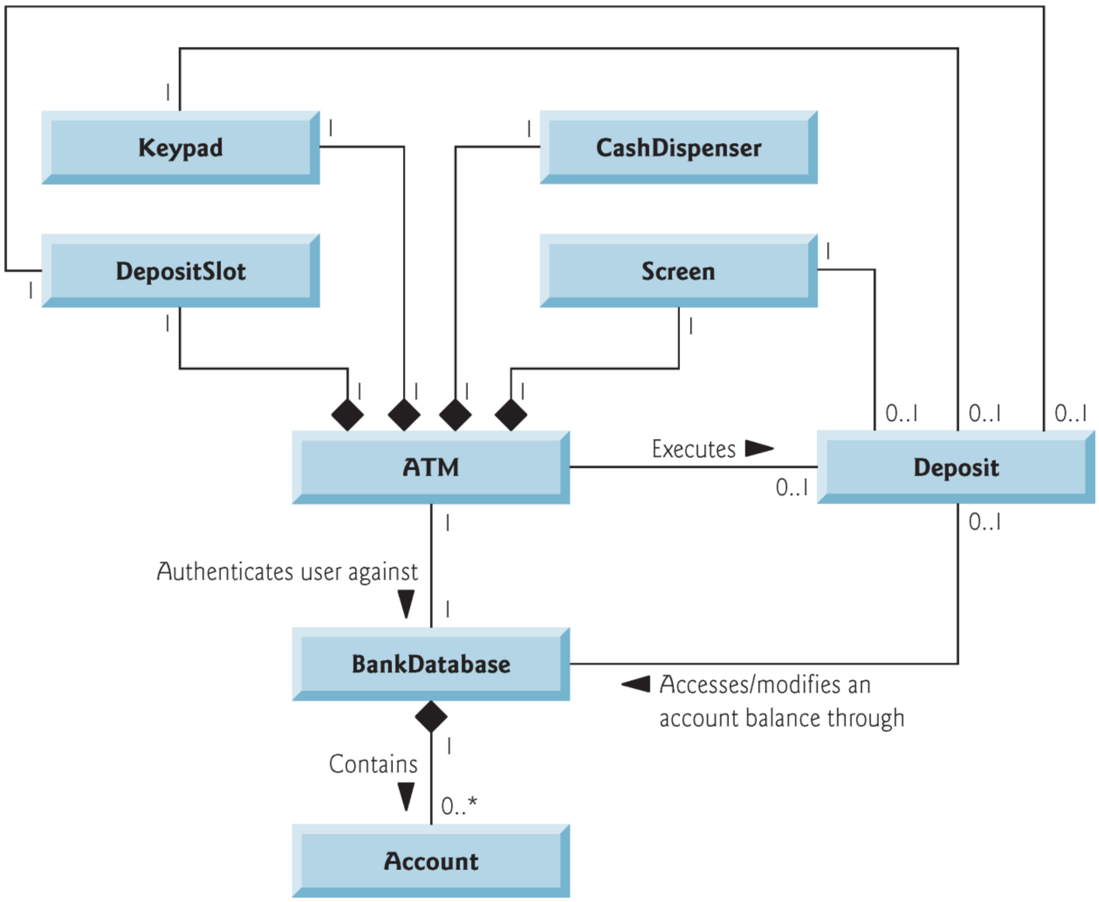

Fig. 12.28 | Class diagram for the ATM system model including class `Deposit`.

(C) 2010 Pearson Education, Inc. All
rights reserved.

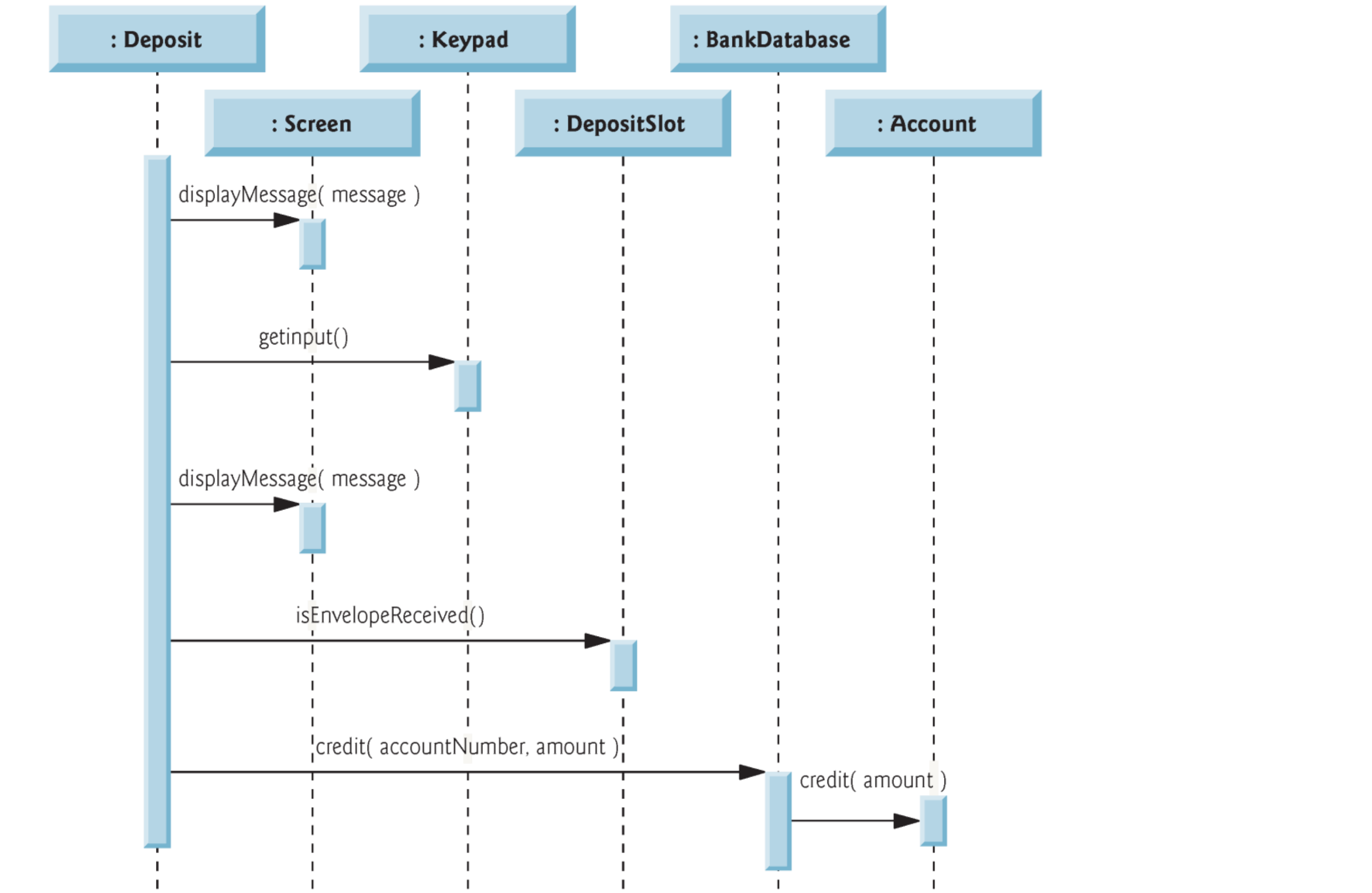

Fig. 12.30 | Sequence diagram that models a `Deposit` executing.

(C) 2010 Pearson Education, Inc. All rights reserved.

Useful material

- Java: How to Program. 9th ed. by Deitel & Deitel
  - Chapters 1- 10
  - This case study: chapters 12 and 13.

(C) 2010 Pearson Education, Inc. All rights reserved.

End of Part I

(C) 2010 Pearson Education, Inc. All rights reserved.