Chapter 26
ATM Case Study, Part 2: Implementing an Object-Oriented Design

C++ How to Program, 9/e

OBJECTIVES

In this chapter you'll:

- Incorporate inheritance into the design of the ATM.
- Incorporate polymorphism into the design of the ATM.
- Fully implement in C++ the UML-based object-oriented design of the ATM software.
- Study a detailed code walkthrough of the ATM software system that explains the implementation issues.

26.2 Starting to Program the Classes of the ATM System

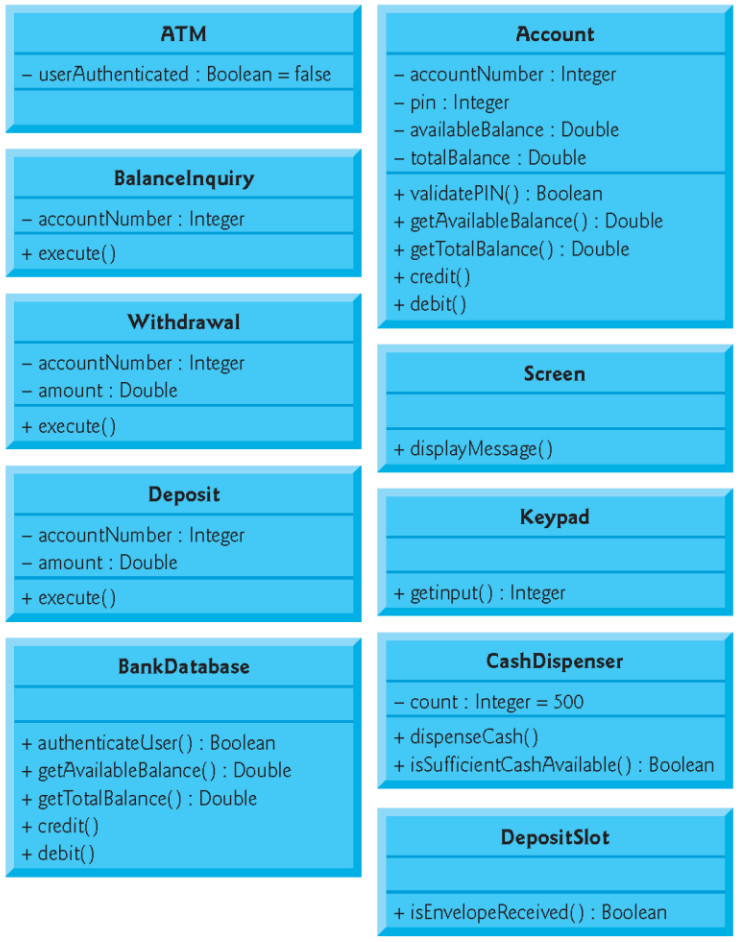
ATM
– userAuthenticated : Boolean = false
BalanceInquiry
– accountNumber : Integer
+ execute()
Withdrawal
– accountNumber : Integer
– amount : Double
+ execute()
Deposit
– accountNumber : Integer
– amount : Double
+ execute()
BankDatabase
+ authenticateUser() : Boolean
+ getAvailableBalance() : Double
+ getTotalBalance() : Double
+ credit()
+ debit()
Account
– accountNumber : Integer
– pin : Integer
– availableBalance : Double
– totalBalance : Double
+ validatePIN() : Boolean
+ getAvailableBalance() : Double
+ getTotalBalance() : Double
+ credit()
+ debit()
Screen
+ displayMessage()
Keypad
+ getinput() : Integer
CashDispenser
– count : Integer = 500
+ dispenseCash()
+ isSufficientCashAvailable() : Boolean
DepositSlot
+ isEnvelopeReceived() : Boolean

Fig. 25.1 | Class diagram with visibility markers.

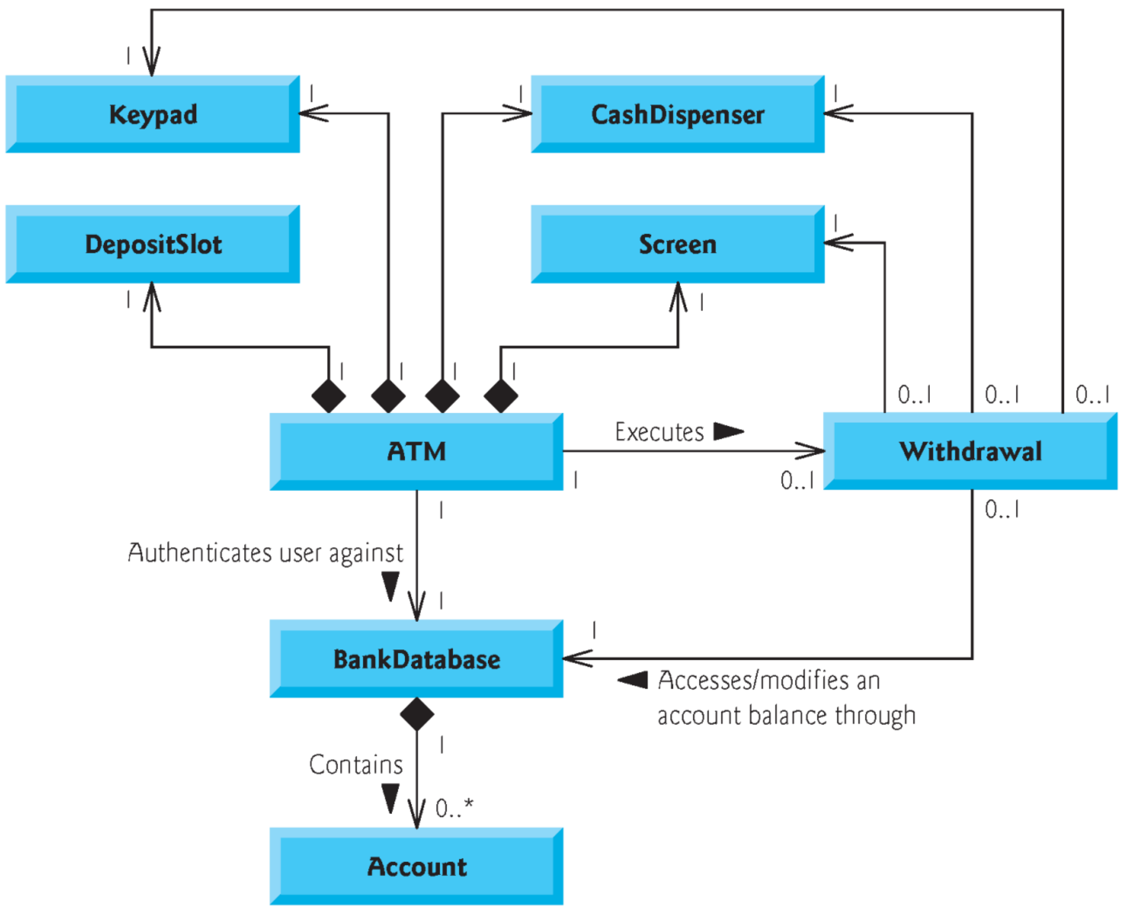

Fig. 25.2 | Class diagram with navigability arrows.

```cpp
// Fig. 26.3: Withdrawal.h
// Definition of class Withdrawal that represents a withdrawal transaction
#ifndef WITHDRAWAL_H
#define WITHDRAWAL_H

class Withdrawal
{
}; // end class Withdrawal

#endif // WITHDRAWAL_H
```

Fig. 25.3 | Definition of class `Withdrawal` enclosed in preprocessor wrappers.

```cpp
// Fig. 26.4: Withdrawal.h
// Definition of class Withdrawal that represents a withdrawal transaction
#ifndef WITHDRAWAL_H
#define WITHDRAWAL_H

class Withdrawal
{
private:
   // attributes
   int accountNumber; // account to withdraw funds from
   double amount; // amount to withdraw
}; // end class Withdrawal

#endif // WITHDRAWAL_H
```

Fig. 25.4 | Adding attributes to the `Withdrawal` class header file.

```cpp
// Fig. 26.5: Withdrawal.h
// Definition of class Withdrawal that represents a withdrawal transaction
#ifndef WITHDRAWAL_H
#define WITHDRAWAL_H

#include "Screen.h" // include definition of class Screen
#include "Keypad.h" // include definition of class Keypad
#include "CashDispenser.h" // include definition of class CashDispenser
#include "BankDatabase.h" // include definition of class BankDatabase

class Withdrawal
{
private:
   // attributes
   int accountNumber; // account to withdraw funds from
   double amount; // amount to withdraw

```

Fig. 25.5 | Declaring references to objects associated with class `Withdrawal`. (Part 1 of 2.)

```
    // references to associated objects
    Screen &screen; // reference to ATM’s screen
    Keypad &keypad; // reference to ATM's keypad
    CashDispenser &cashDispenser; // reference to ATM's cash dispenser
    BankDatabase &bankDatabase; // reference to the account info database
}; // end class Withdrawal

#endif // WITHDRAWAL_H
```

Fig. 25.5 | Declaring references to objects associated with class `Withdrawal`. (Part 2 of 2.)

```cpp
// Fig. 26.6: Withdrawal.h
// Definition of class Withdrawal that represents a withdrawal transaction
#ifndef WITHDRAWAL_H
#define WITHDRAWAL_H

class Screen; // forward declaration of class Screen
class Keypad; // forward declaration of class Keypad
class CashDispenser; // forward declaration of class CashDispenser
class BankDatabase; // forward declaration of class BankDatabase

class Withdrawal
{
private:
   // attributes
   int accountNumber; // account to withdraw funds from
   double amount; // amount to withdraw

```

Fig. 25.6 | Using forward declarations in place of `#include` directives. (Part 1 of 2.)

```
   // references to associated objects
   Screen &screen; // reference to ATM's screen
   Keypad &keypad; // reference to ATM's keypad
   CashDispenser &cashDispenser; // reference to ATM's cash dispenser
   BankDatabase &bankDatabase; // reference to the account info database
}; // end class Withdrawal

#endif // WITHDRAWAL_H
```

Fig. 25.6 | Using forward declarations in place of `#include` directives. (Part 2 of 2.)

Software Engineering Observation 25.1

Several UML modeling tools can convert UML-based designs into C++ code, considerably speeding the implementation process. For more information on these "automatic" code generators, refer to our UML Resource Center at `www.deitel.com/UML/`.

```cpp
// Fig. 26.7: Withdrawal.h
// Definition of class Withdrawal that represents a withdrawal transaction
#ifndef WITHDRAWAL_H
#define WITHDRAWAL_H

class Screen; // forward declaration of class Screen
class Keypad; // forward declaration of class Keypad
class CashDispenser; // forward declaration of class CashDispenser
class BankDatabase; // forward declaration of class BankDatabase

class Withdrawal
{
public:
   // operations
   void execute(); // perform the transaction
private:
   // attributes
   int accountNumber; // account to withdraw funds from
   double amount; // amount to withdraw

```

Fig. 25.7 | Adding operations to the `Withdrawal` class header file. (Part 1 of 2.)

```
    // references to associated objects
    Screen &screen; // reference to ATM's screen
    Keypad &keypad; // reference to ATM's keypad
    CashDispenser &cashDispenser; // reference to ATM's cash dispenser
    BankDatabase &bankDatabase; // reference to the account info database
}; // end class Withdrawal

#endif // WITHDRAWAL_H
```

Fig. 25.7 | Adding operations to the `Withdrawal` class header file. (Part 2 of 2.)