26.3 Incorporating Inheritance into the ATM System

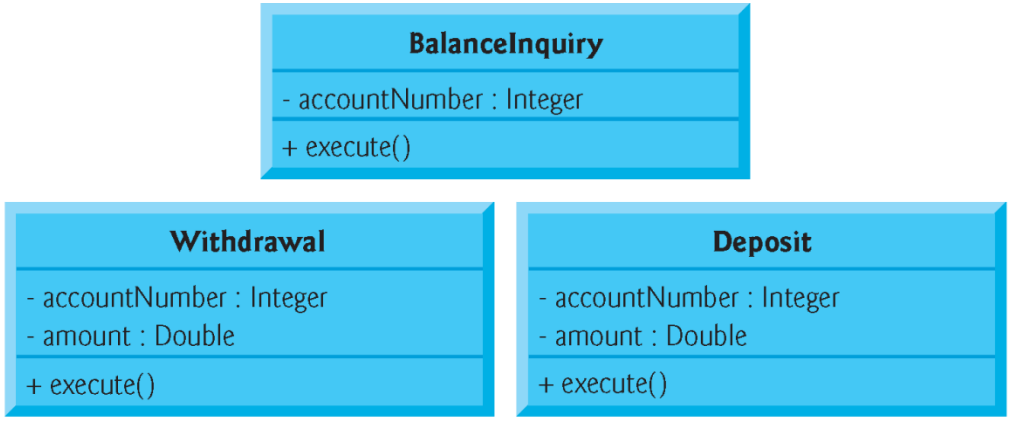

Fig. 25.8 | Attributes and operations of classes `BalanceInquiry`, `Withdrawal` and `Deposit`.

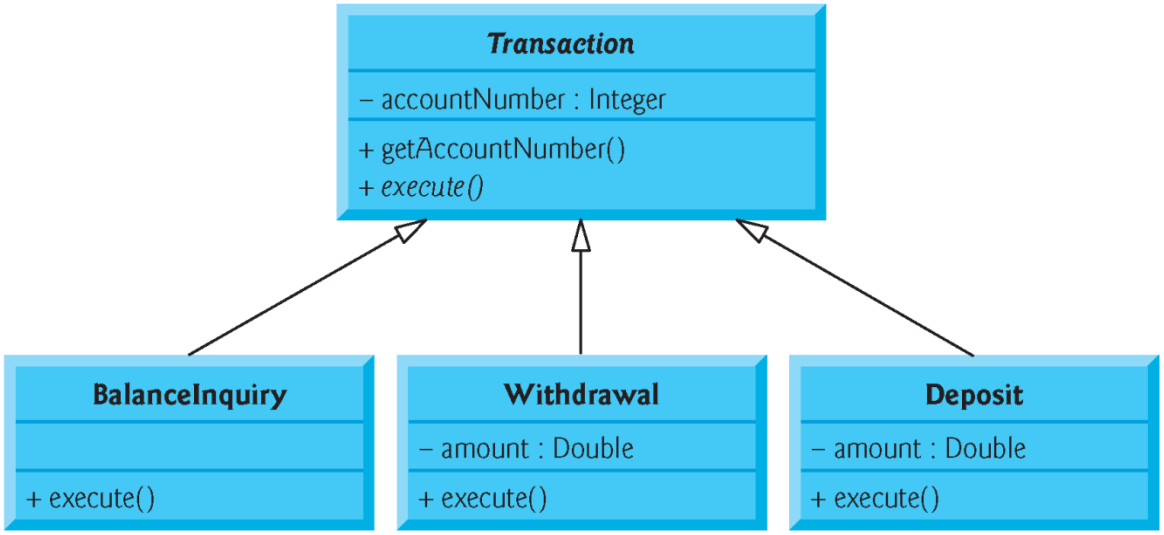

Fig. 25.9 | Class diagram modeling generalization relations\hip between base class `Transaction` and derived classes `BalanceInquiry`, `Withdrawal` and `Deposit`.

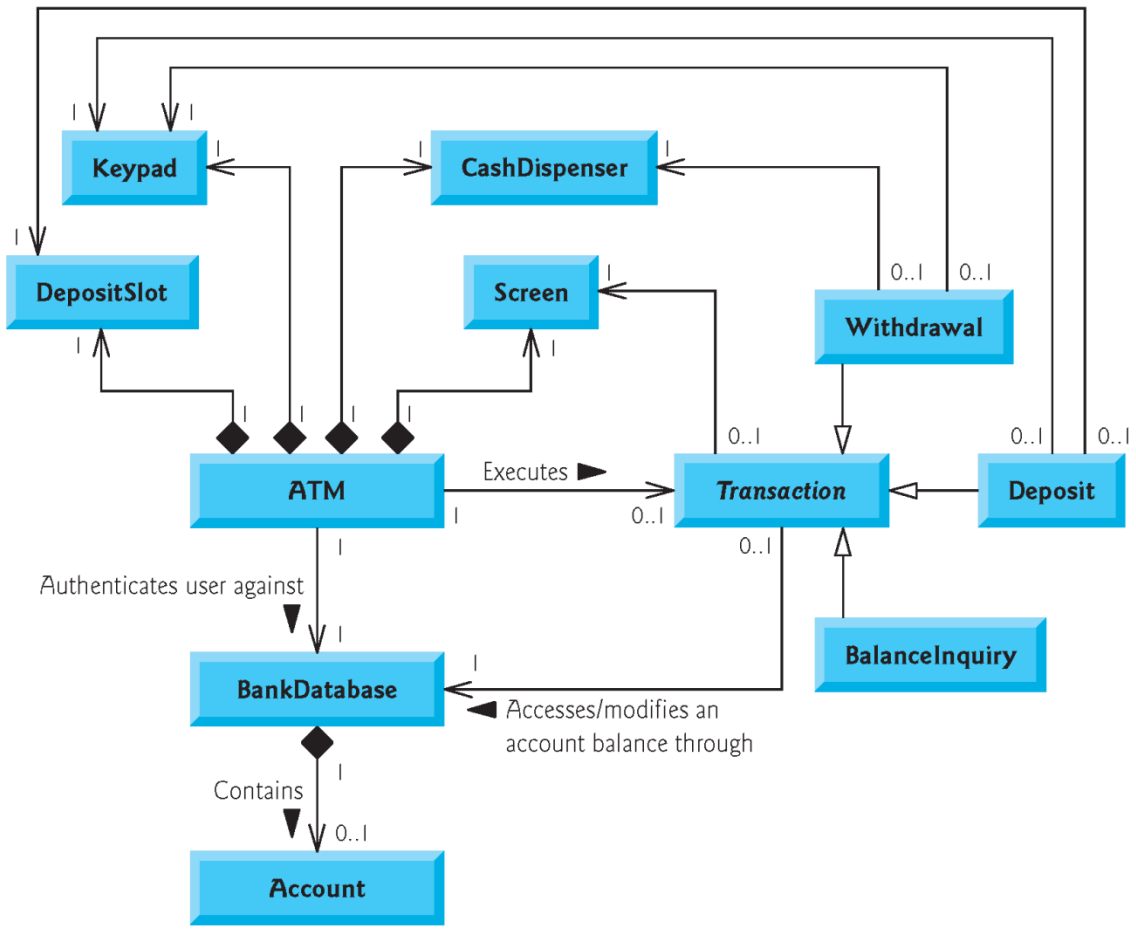

Fig. 25.10 | Class diagram of the ATM system (incorporating inheritance). Note that abstract class name `Transaction` appears in italics.

Fig. 25.11 | Class diagram after incorporating inheritance into the system.

Software Engineering Observation 25.2

A complete class diagram shows all the associations among classes and all the attributes and operations for each class. When the number of class attributes, operations and associations is substantial (as in Fig. 26.10 and Fig. 26.11), a good practice that promotes readability is to divide this information between two class diagrams—one focusing on associations and the other on attributes and operations. However, when examining classes modeled in this fashion, it's crucial to consider both class diagrams to get a complete view of the classes. For example, one must refer to Fig. 26.10 to observe the inheritance relationship between `Transaction` and its derived classes that is omitted from Fig. 26.11.

```
1  // Fig. 26.12: Withdrawal.h
2  // Definition of class Withdrawal that represents a withdrawal transaction
3  #ifndef WITHDRAWAL_H
4  #define WITHDRAWAL_H
5
6  #include "Transaction.h" // Transaction class definition
7
8  // class Withdrawal derives from base class Transaction
9  class Withdrawal : public Transaction
10 {
11 }; // end class Withdrawal
12
13 #endif // WITHDRAWAL_H
```

Fig. 25.12 | `Withdrawal` class definition that derives from `Transaction`.

```
1  // Fig. 26.13: Withdrawal.h
2  // Definition of class Withdrawal that represents a withdrawal transaction
3  #ifndef WITHDRAWAL_H
4  #define WITHDRAWAL_H
5
6  #include "Transaction.h" // Transaction class definition
7
8  class Keypad; // forward declaration of class Keypad
9  class CashDispenser; // forward declaration of class CashDispenser
10
11 // class Withdrawal derives from base class Transaction
12 class Withdrawal : public Transaction
13 {
14 public:
15    // member function overriding execute in base class Transaction
16    virtual void execute(); // perform the transaction
17 private:
18    // attributes
19    double amount; // amount to withdraw
20    Keypad &keypad; // reference to ATM's keypad
21    CashDispenser &cashDispenser; // reference to ATM's cash dispenser
22 }; // end class Withdrawal
23
24 #endif // WITHDRAWAL_H
```

Fig. 25.13 | `Withdrawal` class header file based on Fig. 26.10 and Fig. 26.11.

26.4 ATM Case Study Implementation

26.4.1 Class ATM

```
1  // ATM.h
2  // ATM class definition. Represents an automated teller machine.
3  #ifndef ATM_H
4  #define ATM_H
5
6  #include "Screen.h" // Screen class definition
7  #include "Keypad.h" // Keypad class definition
8  #include "CashDispenser.h" // CashDispenser class definition
9  #include "DepositSlot.h" // DepositSlot class definition
10 #include "BankDatabase.h" // BankDatabase class definition
11 class Transaction; // forward declaration of class Transaction
12
13 class ATM
14 {
15 public:
16    ATM(); // constructor initializes data members
17    void run(); // start the ATM
18 private:
19    bool userAuthenticated; // whether user is authenticated
20    int currentAccountNumber; // current user's account number
21    Screen screen; // ATM's screen
22    Keypad keypad; // ATM's keypad
23    CashDispenser cashDispenser; // ATM's cash dispenser
24    DepositSlot depositSlot; // ATM's deposit slot
```

Fig. 25.14 | Definition of class `ATM`, which represents the ATM. (Part 1 of 2.)

```
25      BankDatabase bankDatabase; // account information database
26
27      // private utility functions
28      void authenticateUser(); // attempts to authenticate user
29      void performTransactions(); // performs transactions
30      int displayMainMenu() const; // displays main menu
31
32      // return object of specified Transaction derived class
33      Transaction *createTransaction( int );
34  }; // end class ATM
35
36  #endif // ATM_H
```

Fig. 25.14 | Definition of class `ATM`, which represents the ATM. (Part 2 of 2.)

```
1  // ATM.cpp
2  // Member-function definitions for class ATM.
3  #include "ATM.h" // ATM class definition
4  #include "Transaction.h" // Transaction class definition
5  #include "BalanceInquiry.h" // BalanceInquiry class definition
6  #include "Withdrawal.h" // Withdrawal class definition
7  #include "Deposit.h" // Deposit class definition
8
9  // enumeration constants represent main menu options
10 enum MenuOption { BALANCE_INQUIRY = 1, WITHDRAWAL, DEPOSIT, EXIT };
11
12 // ATM default constructor initializes data members
13 ATM::ATM()
14    : userAuthenticated ( false ), // user is not authenticated to start
15      currentAccountNumber( 0 ) // no current account number to start
16 {
17    // empty body
18 } // end ATM default constructor
19
```

Fig. 25.15 | ATM class member-function definitions. (Part 1 of 7.)

```
20  // start ATM
21  void ATM::run()
22  {
23     // welcome and authenticate user; perform transactions
24     while ( true )
25     {
26        // loop while user is not yet authenticated
27        while ( !userAuthenticated )
28        {
29           screen.displayMessageLine( "\nWelcome!" );
30           authenticateUser(); // authenticate user
31        } // end while
32
33        performTransactions(); // user is now authenticated
34        userAuthenticated = false; // reset before next ATM session
35        currentAccountNumber = 0; // reset before next ATM session
36        screen.displayMessageLine( "\nThank you! Goodbye!" );
37     } // end while
38  } // end function run
39
```

Fig. 25.15 | ATM class member-function definitions. (Part 2 of 7.)

```
40  // attempt to authenticate user against database
41  void ATM::authenticateUser()
42  {
43     screen.displayMessage( "\nPlease enter your account number: " );
44     int accountNumber = keypad.getInput(); // input account number
45     screen.displayMessage( "\nEnter your PIN: " ); // prompt for PIN
46     int pin = keypad.getInput(); // input PIN
47
48     // set userAuthenticated to bool value returned by database
49     userAuthenticated =
50        bankDatabase.authenticateUser( accountNumber, pin );
51
52     // check whether authentication succeeded
53     if ( userAuthenticated )
54     {
55        currentAccountNumber = accountNumber; // save user's account #
56     } // end if
57     else
58        screen.displayMessageLine(
59           "Invalid account number or PIN. Please try again." );
60  } // end function authenticateUser
61
```

Fig. 25.15 | ATM class member-function definitions. (Part 3 of 7.)