```
// display the main menu and perform transactions
void ATM::performTransactions()
{
   // local pointer to store transaction currently being processed
   Transaction *currentTransactionPtr;

   bool userExited = false; // user has not chosen to exit

   // loop while user has not chosen option to exit system
   while ( !userExited )
   {
      // show main menu and get user selection
      int mainMenuSelection = displayMainMenu();

      // decide how to proceed based on user's menu selection
      switch ( mainMenuSelection )
      {
         // user chose to perform one of three transaction types
         case BALANCE_INQUIRY:
         case WITHDRAWAL:
         case DEPOSIT:
            // initialize as new object of chosen type
            currentTransactionPtr =
               createTransaction( mainMenuSelection );
```

Fig. 25.15 | ATM class member-function definitions. (Part 4 of 7.)

```cpp

               currentTransactionPtr->execute(); // execute transaction

               // free the space for the dynamically allocated Transaction
               delete currentTransactionPtr;

               break;
            case EXIT: // user chose to terminate session
               screen.displayMessageLine( "\nExiting the system..." );
               userExited = true; // this ATM session should end
               break;
            default: // user did not enter an integer from 1-4
               screen.displayMessageLine(
                  "\nYou did not enter a valid selection. Try again." );
               break;
         } // end switch
      } // end while
   } // end function performTransactions

```

Fig. 25.15 | ATM class member-function definitions. (Part 5 of 7.)

```
// display the main menu and return an input selection
int ATM::displayMainMenu() const
{
   screen.displayMessageLine( "\nMain menu:" );
   screen.displayMessageLine( "1 - View my balance" );
   screen.displayMessageLine( "2 - Withdraw cash" );
   screen.displayMessageLine( "3 - Deposit funds" );
   screen.displayMessageLine( "4 - Exit\n" );
   screen.displayMessage( "Enter a choice: " );
   return keypad.getInput(); // return user's selection
} // end function displayMainMenu

// return object of specified Transaction derived class
Transaction *ATM::createTransaction( int type )
{
   Transaction *tempPtr; // temporary Transaction pointer

```

Fig. 25.15 | ATM class member-function definitions. (Part 6 of 7.)

```
   // determine which type of Transaction to create
   switch ( type )
   {
      case BALANCE_INQUIRY: // create new BalanceInquiry transaction
         tempPtr = new BalanceInquiry(
            currentAccountNumber, screen, bankDatabase );
         break;
      case WITHDRAWAL: // create new Withdrawal transaction
         tempPtr = new Withdrawal( currentAccountNumber, screen,
            bankDatabase, keypad, cashDispenser );
         break;
      case DEPOSIT: // create new Deposit transaction
         tempPtr = new Deposit( currentAccountNumber, screen,
            bankDatabase, keypad, depositSlot );
         break;
   } // end switch

   return tempPtr; // return the newly created object
} // end function createTransaction
```

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

26.4.2 Class Screen

```cpp
// Screen.h
// Screen class definition. Represents the screen of the ATM.
#ifndef SCREEN_H
#define SCREEN_H

#include <string>
using namespace std;

class Screen
{
public:
   void displayMessage( string ) const; // output a message
   void displayMessageLine( string ) const; // output message with newline
   void displayDollarAmount( double ) const; // output a dollar amount
}; // end class Screen

#endif // SCREEN_H
```

Fig. 25.16 | Screen class definition.

```cpp
// Screen.cpp
// Member-function definitions for class Screen.
#include <iostream>
#include <iomanip>
#include "Screen.h" // Screen class definition
using namespace std;

// output a message without a newline
void Screen::displayMessage( string message ) const
{
   cout << message;
} // end function displayMessage

// output a message with a newline
void Screen::displayMessageLine( string message ) const
{
   cout << message << endl;
} // end function displayMessageLine

// output a dollar amount
void Screen::displayDollarAmount( double amount ) const
{
   cout << fixed << setprecision( 2 ) << "$" << amount;
} // end function displayDollarAmount
```

Fig. 25.17 | Screen class member-function definitions.

26.4.3 Class Keypad

```cpp
// Keypad.h
// Keypad class definition. Represents the keypad of the ATM.
#ifndef KEYPAD_H
#define KEYPAD_H

class Keypad
{
public:
   int getInput() const; // return an integer value entered by user
}; // end class Keypad

#endif // KEYPAD_H
```

Fig. 25.18 | Keypad class definition.

```cpp
// Keypad.cpp
// Member-function definition for class Keypad (the ATM's keypad).
#include <iostream>
using namespace std;

#include "Keypad.h" // Keypad class definition

// return an integer value entered by user
int Keypad::getInput() const
{
   int input; // variable to store the input
   cin >> input; // we assume that user enters an integer
   return input; // return the value entered by user
} // end function getInput
```

Fig. 25.19 | Keypad class member-function definition.

26.4.4 Class `CashDispenser`

```cpp
// CashDispenser.h
// CashDispenser class definition. Represents the ATM's cash dispenser.
#ifndef CASH_DISPENSER_H
#define CASH_DISPENSER_H

class CashDispenser
{
public:
   CashDispenser(); // constructor initializes bill count to 500

   // simulates dispensing of specified amount of cash
   void dispenseCash( int );

   // indicates whether cash dispenser can dispense desired amount
   bool isSufficientCashAvailable( int ) const;
private:
   static const int INITIAL_COUNT = 500;
   int count; // number of $20 bills remaining
}; // end class CashDispenser

#endif // CASH_DISPENSER_H
```

Fig. 25.20 | `CashDispenser` class definition.

```cpp
// CashDispenser.cpp
// Member-function definitions for class CashDispenser.
#include "CashDispenser.h" // CashDispenser class definition

// CashDispenser default constructor initializes count to default
CashDispenser::CashDispenser()
{
   count = INITIAL_COUNT; // set count attribute to default
} // end CashDispenser default constructor

// simulates dispensing of specified amount of cash; assumes enough cash
// is available (previous call to isSufficientCashAvailable returned true)
void CashDispenser::dispenseCash( int amount )
{
   int billsRequired = amount / 20; // number of $20 bills required
   count -= billsRequired; // update the count of bills
} // end function dispenseCash

```

Fig. 25.21 | `CashDispenser` class member-function definitions. (Part 1 of 2.)

```cpp
// indicates whether cash dispenser can dispense desired amount
bool CashDispenser::isSufficientCashAvailable( int amount ) const
{
   int billsRequired = amount / 20; // number of $20 bills required

   if ( count >= billsRequired )
      return true; // enough bills are available
   else
      return false; // not enough bills are available
} // end function isSufficientCashAvailable
```

Fig. 25.21 | `CashDispenser` class member-function definitions. (Part 2 of 2.)

26.4.5 Class `DepositSlot`