```cpp
// Transaction.h
// Transaction abstract base class definition.
#ifndef TRANSACTION_H
#define TRANSACTION_H

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

class Transaction
{
public:
   // constructor initializes common features of all Transactions
   Transaction( int, Screen &, BankDatabase & );

   virtual ~Transaction() { } // virtual destructor with empty body

   int getAccountNumber() const; // return account number
   Screen &getScreen() const; // return reference to screen
   BankDatabase &getBankDatabase() const; // return reference to database

```

Fig. 25.28 | Transaction class definition. (Part 1 of 2.)

```cpp
   // pure virtual function to perform the transaction
   virtual void execute() = 0; // overridden in derived classes
private:
   int accountNumber; // indicates account involved
   Screen &screen; // reference to the screen of the ATM
   BankDatabase &bankDatabase; // reference to the account info database
}; // end class Transaction

#endif // TRANSACTION_H
```

Fig. 25.28 | Transaction class definition. (Part 2 of 2.)

```cpp
// Transaction.cpp
// Member-function definitions for class Transaction.
#include "Transaction.h" // Transaction class definition
#include "Screen.h" // Screen class definition
#include "BankDatabase.h" // BankDatabase class definition

// constructor initializes common features of all Transactions
Transaction::Transaction( int userAccountNumber, Screen &atmScreen,
   BankDatabase &atmBankDatabase )
   : accountNumber( userAccountNumber ),
     screen( atmScreen ),
     bankDatabase( atmBankDatabase )
{
   // empty body
} // end Transaction constructor

// return account number
int Transaction::getAccountNumber() const
{
   return accountNumber;
} // end function getAccountNumber

```

Fig. 25.29 | Transaction class member-function definitions. (Part 1 of 2.)

```cpp
// return reference to screen
Screen &Transaction::getScreen() const
{
   return screen;
} // end function getScreen

// return reference to bank database
BankDatabase &Transaction::getBankDatabase() const
{
   return bankDatabase;
} // end function getBankDatabase
```

Fig. 25.29 | `Transaction` class member-function definitions. (Part 2 of 2.)

26.4.9 Class `BalanceInquiry`

```cpp
// BalanceInquiry.h
// BalanceInquiry class definition. Represents a balance inquiry.
#ifndef BALANCE_INQUIRY_H
#define BALANCE_INQUIRY_H

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

class BalanceInquiry : public Transaction
{
public:
   BalanceInquiry( int, Screen &, BankDatabase & ); // constructor
   virtual void execute(); // perform the transaction
}; // end class BalanceInquiry

#endif // BALANCE_INQUIRY_H
```

Fig. 25.30 | `BalanceInquiry` class definition.

```cpp
// BalanceInquiry.cpp
// Member-function definitions for class BalanceInquiry.
#include "BalanceInquiry.h" // BalanceInquiry class definition
#include "Screen.h" // Screen class definition
#include "BankDatabase.h" // BankDatabase class definition

// BalanceInquiry constructor initializes base-class data members
BalanceInquiry:: BalanceInquiry( int userAccountNumber, Screen &atmScreen,
   BankDatabase &atmBankDatabase )
   : Transaction( userAccountNumber, atmScreen, atmBankDatabase )
{
   // empty body
} // end BalanceInquiry constructor

// performs transaction; overrides Transaction's pure virtual function
void BalanceInquiry::execute()
{
   // get references to bank database and screen
   BankDatabase &bankDatabase = getBankDatabase();
   Screen &screen = getScreen();

```

Fig. 25.31 | BalanceInquiry class member-function definitions. (Part 1 of 2.)

```
      // get the available balance for the current user's Account
      double availableBalance =
         bankDatabase.getAvailableBalance( getAccountNumber() );

      // get the total balance for the current user's Account
      double totalBalance =
         bankDatabase.getTotalBalance( getAccountNumber() );

      // display the balance information on the screen
      screen.displayMessageLine( "\nBalance Information:" );
      screen.displayMessage( " - Available balance: " );
      screen.displayDollarAmount( availableBalance );
      screen.displayMessage( "\n - Total balance:     " );
      screen.displayDollarAmount( totalBalance );
      screen.displayMessageLine( "" );
   } // end function execute
```

Fig. 25.31 | BalanceInquiry class member-function definitions. (Part 2 of 2.)

26.4.10 Class `Withdrawal`

```cpp
// Withdrawal.h
// Withdrawal class definition. Represents a withdrawal transaction.
#ifndef WITHDRAWAL_H
#define WITHDRAWAL_H

#include "Transaction.h" // Transaction class definition
class Keypad; // forward declaration of class Keypad
class CashDispenser; // forward declaration of class CashDispenser

class Withdrawal : public Transaction
{
public:
   Withdrawal( int, Screen &, BankDatabase &, Keypad &, CashDispenser & );
   virtual void execute(); // perform the transaction
private:
   int amount; // amount to withdraw
   Keypad &keypad; // reference to ATM's keypad
   CashDispenser &cashDispenser; // reference to ATM's cash dispenser
   int displayMenuOfAmounts() const; // display the withdrawal menu
}; // end class Withdrawal

#endif // WITHDRAWAL_H
```

Fig. 25.32 | `Withdrawal` class definition.

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

// global constant that corresponds to menu option to cancel
static const int CANCELED = 6;

// Withdrawal constructor initialize class's data members
Withdrawal::Withdrawal( int userAccountNumber, Screen &atmScreen,
   BankDatabase &atmBankDatabase, Keypad &atmKeypad,
   CashDispenser &atmCashDispenser )
   : Transaction( userAccountNumber, atmScreen, atmBankDatabase ),
     keypad( atmKeypad ), cashDispenser( atmCashDispenser )
{
   // empty body
} // end Withdrawal constructor

```

Fig. 25.33 | Withdrawal class member-function definitions. (Part 1 of 6.)

```
// perform transaction; overrides Transaction's pure virtual function
void Withdrawal::execute()
{
   bool cashDispensed = false; // cash was not dispensed yet
   bool transactionCanceled = false; // transaction was not canceled yet

   // get references to bank database and screen
   BankDatabase &bankDatabase = getBankDatabase();
   Screen &screen = getScreen();

   // loop until cash is dispensed or the user cancels
   do
   {
      // obtain the chosen withdrawal amount from the user
      int selection = displayMenuOfAmounts();

      // check whether user chose a withdrawal amount or canceled
      if ( selection != CANCELED )
      {
         amount = selection; // set amount to the selected dollar amount

         // get available balance of account involved
         double availableBalance =
            bankDatabase.getAvailableBalance( getAccountNumber() );
```

Fig. 25.33 | `Withdrawal` class member-function definitions. (Part 2 of 6.)

```

      // check whether the user has enough money in the account
      if ( amount <= availableBalance )
      {
         // check whether the cash dispenser has enough money
         if ( cashDispenser.isSufficientCashAvailable( amount ) )
         {
            // update the account involved to reflect withdrawal
            bankDatabase.debit( getAccountNumber(), amount );

            cashDispenser.dispenseCash( amount ); // dispense cash
            cashDispensed = true; // cash was dispensed

            // instruct user to take cash
            screen.displayMessageLine(
               "\nPlease take your cash from the cash dispenser." );
         } // end if
         else // cash dispenser does not have enough cash
            screen.displayMessageLine(
               "\nInsufficient cash available in the ATM."
               "\n\nPlease choose a smaller amount." );
      } // end if
```

Fig. 25.33 | `Withdrawal` class member-function definitions. (Part 3 of 6.)

```
            else // not enough money available in user's account
            {
               screen.displayMessageLine(
                  "\nInsufficient funds in your account."
                  "\n\nPlease choose a smaller amount." );
            } // end else
         } // end if
         else // user chose cancel menu option
         {
            screen.displayMessageLine( "\nCanceling transaction..." );
            transactionCanceled = true; // user canceled the transaction
         } // end else
      } while ( !cashDispensed && !transactionCanceled ); // end do...while
} // end function execute

// display a menu of withdrawal amounts and the option to cancel;
// return the chosen amount or 0 if the user chooses to cancel
int Withdrawal::displayMenuOfAmounts() const
{
   int userChoice = 0; // local variable to store return value

   Screen &screen = getScreen(); // get screen reference

```

Fig. 25.33 | Withdrawal class member-function definitions. (Part 4 of 6.)

```
      // array of amounts to correspond to menu numbers
      int amounts[] = { 0, 20, 40, 60, 100, 200 };

      // loop while no valid choice has been made
      while ( userChoice == 0 )
      {
         // display the menu
         screen.displayMessageLine( "\nWithdrawal options:" );
         screen.displayMessageLine( "1 - $20" );
         screen.displayMessageLine( "2 - $40" );
         screen.displayMessageLine( "3 - $60" );
         screen.displayMessageLine( "4 - $100" );
         screen.displayMessageLine( "5 - $200" );
         screen.displayMessageLine( "6 - Cancel transaction" );
         screen.displayMessage( "\nChoose a withdrawal option (1-6): " );

         int input = keypad.getInput(); // get user input through keypad

```

Fig. 25.33 | `Withdrawal` class member-function definitions. (Part 5 of 6.)