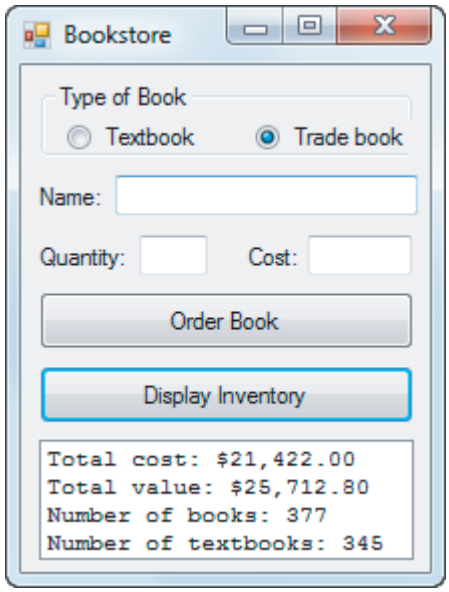

FIGURE 11.9 Sample output for Exercise 21.

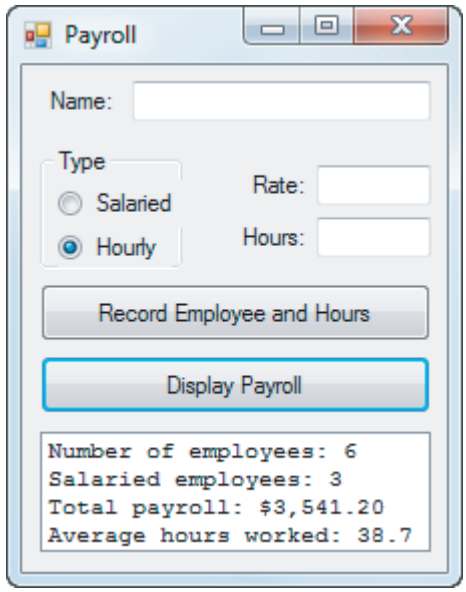

FIGURE 11.10 Sample output for Exercise 22.

class and override the method GrossPay that accepts the number of hours worked as a parameter. A sample output is shown in Fig. 11.10. (***Hint:*** Use an array of a structure that holds the employee object and the number of hours worked during the week.)

Solutions to Practice Problems 11.3

1. While the derived class Calculator has access to the Properties and Methods of the base class AddingMachine, it does not have access to its Private member variables.
2. The string "Can move Has a backbone Nurtures young with mother's milk"

CHAPTER 11 SUMMARY

1. An *object* is an entity that stores data, has methods that manipulate the data, and can raise events. A *class* is a template from which objects are created. A *method* specifies the way in which an object's data are manipulated. An *event* is a message sent by an object to signal the occurrence of a condition.
2. Each class is defined in a separate block of code starting with Class *ClassName* and ending with End Class. Data are stored in member variables and accessed by procedures called properties.
3. A property routine contains a Get block to retrieve the value of a member variable or a Set block to assign a value to a member variable. These procedures can also be used to enforce constraints and carry out validation.
4. Visual Basic automatically invokes a New procedure when an object is created.
5. An object variable is declared with a statement of the form `Dim objectName As ClassName`, and the object is created with a statement of the form `objectName = New ClassName(arg1, arg2, ...)`. These two statements are often combined into the single statement `Dim objectName As New ClassName(arg1, arg2, ...)`.
6. *Auto-implemented properties* enable you to quickly specify a property of a class without having to write code to Get and Set the property.

7. Events are declared in the Declarations section of a class with a statement of the form `Public Event UserDefinedEvent(arg1, arg2, ...)` and raised with a `RaiseEvent` statement. The declaration statement for the object must include the keyword `WithEvents` in order for the events coming from the object to be processed. The header of an event-handling procedure has the form `Private Sub procedureName(par1, par2, ...) Handles objectName.UserDefinedEvent`.

8. The properties, methods, and events of a class are referred to as its *interface*.

9. *Inheritance*, which is implemented with the keyword Inherits, allows a new class (called the *derived* or *child* class) to be created from an existing class (called the *base* or *parent* class) and to gain its interface.

10. *Polymorphism* is the feature that two classes can have methods that are named the same and have essentially the same purpose, but different implementations.

11. The keywords *Overridable*, *Overrides*, *MustInherit*, and *MustOverride* allow derived classes to customize inherited properties and methods.

CHAPTER 11 PROGRAMMING PROJECTS

1. *Bank Account*. Write a program to maintain a person's Savings and Checking accounts. The program should keep track of and display the balances in both accounts, and maintain a list of transactions (deposits, withdrawals, fund transfers, and check clearings) separately for each account. The two lists of transactions should be stored in text files.

   Consider the form in Fig. 11.11. The two drop-down combo boxes should each contain the items Checking and Savings. Each of the four group boxes corresponds to a type of transaction. (When Savings is selected in the Account combo box, the Check group box should disappear.) The user makes a transaction by typing data into the text boxes of a group box and pressing the associated button. The items appearing in the DataGridView

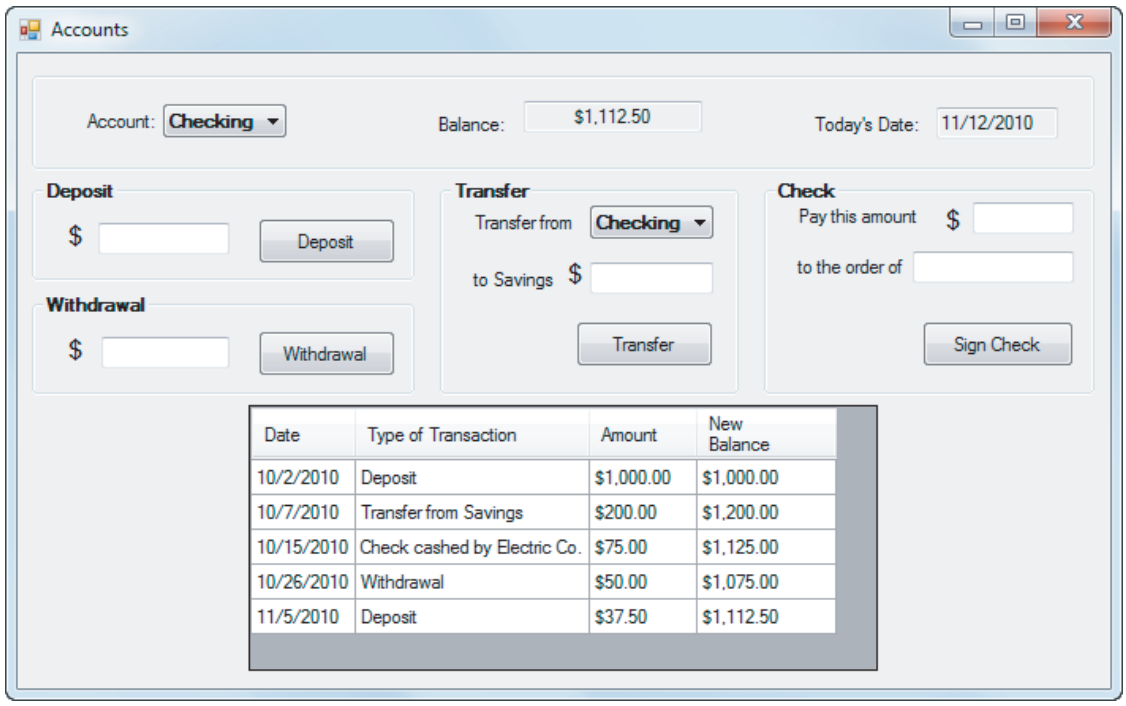

FIGURE 11.11 Bank accounts.

control should correspond to the type of account that has been selected. The caption of the second label in the Transfer group box should toggle between "to Checking" and "to Savings" depending on the item selected in the "Transfer from" combo box. If a transaction cannot be carried out, a message (such as "Insufficient funds") should be displayed. Two text files should be maintained (one for each type of account) and should be updated each time a transaction is carried out.

The program should use two classes, Transaction and Account. The class Transaction should have properties for transaction name, amount, date, and whether it is a credit (deposit) or debit (withdrawal/check).

The class Account, which will have both a checking account and a savings account as instances, should use an array of Transaction objects. In addition, it should have properties for name (Checking or Savings) and balance. It should have methods to carry out a transaction (if possible) and to load the set of transactions from a text file. The events InsufficientFunds and TransactionCommitted should be raised at appropriate times.

2. Write a program for the game BlackJack. See Fig. 11.12. The program should use a DeckOfCards class similar to the one presented in Example 3 of Section 11.2.

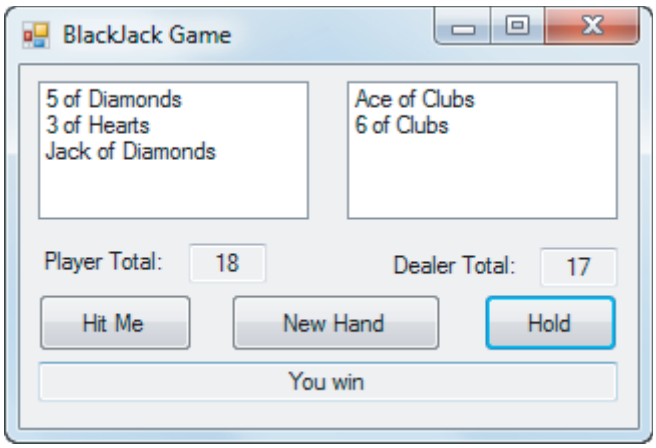

FIGURE 11.12 **Sample output for Programming Project 2.**