Chapter 7

1. **(Card Shuffling and Dealing)**

Modify the application of Fig. 7.13 to deal a five-card poker hand. Then modify class DeckOfCards of Fig. 7.12 to include methods that determine whether a hand contains

a) a pair

b) two pairs

c) three of a kind (e.g., three jacks)

d) four of a kind (e.g., four aces)

e) a flush (i.e., all five cards of the same suit)

f) a straight (i.e., five cards of consecutive face values)

g) a full house (i.e., two cards of one face value and three cards of another face value) [Hint: Add methods getFace and getSuit to class Card of Fig. 7.11.]

**2. (Card Shuffling and Dealing)**

Use the methods developed in Exercise 1 to write an application that deals two five-card poker hands, evaluates each hand and determines which is better.

**3. (Project: Card Shuffling and Dealing)**

Modify the application developed in Exercise 2 so that it can simulate the dealer. The dealer’s five-card hand is dealt “face down,” so the player cannot see it. The application should then evaluate the dealer’s hand, and, based on the quality of the hand, the dealer should draw one, two or three more cards to replace the corresponding number of unneeded cards in the original hand. The application should then reevaluate the dealer’s hand. [Caution: This is a difficult problem!]

**4. (Project: Card Shuffling and Dealing)**

Modify the application developed in Exercise 3 so that it can handle the dealer’s hand automatically, but the player is allowed to decide which cards of the player’s hand to replace. The application should then evaluate both hands and determine who wins. Now use this new application to play 20 games against the computer. Who wins more games, you or the computer? Have a friend play 20 games against the computer. Who wins more games? Based on the results of these games, refine your poker-playing application. (This, too, is a difficult problem.) Play 20 more games. Does your modified application play a better game?

**7.20 (Project: Card Shuffling and Dealing)**

Modify the application of Figs. 7.11–7.13 to use Face and Suit enum types to represent the faces and suits of the cards. Declare each of these enum types as a public type in its own source-code file. Each Card should have a Face and a Suit instance variable. These should be initialized by the Card constructor. In class DeckOfCards, create an array of Faces that’s initialized with the names of the constants in the Face enum type and an array of Suits that’s initialized with the names of the constants in the Suit enum type. [Note: When you output an enum constant as a String, the name of the constant is displayed.]

**7.21 (Fisher-Yates Shuffling Algorithm)**

Research the Fisher-Yates shuffling algorithm online, then use it to reimplement the shuffle method in Fig. 7.12.

**Making a Difference**

7.22 (Target-Heart-Rate Calculator) While exercising, you can use a heart-rate monitor to see that your heart rate stays within a safe range suggested by your trainers and doctors. According to the American Heart Association (AHA) (www.americanheart.org/presenter.jhtml?identifier =4736), the formula for calculating your maximum heart rate in beats per minute is 220 minus your age in years. Your target heart rate is a range that’s 50–85% of your maximum heart rate. [Note: These formulas are estimates provided by the AHA. Maximum and target heart rates may vary based on the health, fitness and gender of the individual. Always consult a physician or qualified healthcare professional before beginning or modifying an exercise program.] Create a class called HeartRates. The class attributes should include the person’s first name, last name and date of birth (consisting of separate attributes for the month, day and year of birth). Your class should have a constructor that receives this data as parameters. For each attribute provide set and get methods. The class also should include a method that calculates and returns the person’s age (in years), a method that calculates and returns the person’s maximum heart rate and a method that calculates and returns the person’s target heart rate. Write a Java app that prompts for the person’s information, instantiates an object of class HeartRates and prints the information from that object—including the person’s first name, last name and date of birth—then calculates and prints the person’s age in (years), maximum heart rate and target-heart-rate range.