1. Playing cards are used in many computer games, including versions of such classics

as solitaire, hearts, and poker. Design a Card class that contains a character data field

to hold a suit (s for spades, h for hearts, d for diamonds, or c for clubs) and an

integer data field for a value from 1 to 13. Include get and set methods for each field. Save the class as Card.java.

Write an application that randomly selects two playing cards and displays their

values. Simply assign a suit to each of the cards, but generate a random number for

each card’s value. Observe how the card values change as you execute the program multiple times. Save the application as PickTwoCards.java.

2. Computer games often contain different characters or creatures. For example, you

might design a game in which alien beings possess specific characteristics such as

color, number of eyes, or number of lives. Design a character for a game, creating a

class to hold at least three attributes for the character. Include methods to get and

set each of the character’s attributes. Save the file as MyCharacter.java. Then write

an application in which you create at least two characters. In turn, pass each

character to a display method that displays the character’s attributes. Save the

application as TwoCharacters.java.

3. Create a class named Billing that includes three overloaded computeBill() methods

for a photo book store.

\* When computeBill() receives a single parameter, it represents the price of one

photo book ordered. Add 8% tax, and return the total due.

\* When computeBill() receives two parameters, they represent the price of a

photo book and the quantity ordered. Multiply the two values, add 8% tax, and

return the total due.

\* When computeBill() receives three parameters, they represent the price of a

photo book, the quantity ordered, and a coupon value. Multiply the quantity and

price, reduce the result by the coupon value, and then add 8% tax and return the

total due.

Write a main() method that tests all three overloaded methods. Save the application

as Billing.java.

4. a. Create a class named BloodData that includes fields that hold a blood type (the

four blood types are O, A, B, and AB) and an Rh factor (the factors are + and –).

Create a default constructor that sets the fields to “O” and “+”, and an overloaded

constructor that requires values for both fields. Include get and set methods for

each field. Save this file as BloodData.java. Create an application named

TestBloodData that demonstrates each method works correctly. Save the

application as TestBloodData.java.

b. Create a class named Patient that includes an ID number, age, and BloodData.

Provide a default constructor that sets the ID number to “0”, the age to 0, and the

BloodData to “O” and “+”. Create an overloaded constructor that provides values

for each field. Also provide get methods for each field. Save the file as Patient.java.

Create an application that demonstrates that each method works correctly, and

save it as TestPatient.java.