Lab 4 –Objects and Classes

Answer the following questions.

Instructor-led Demo:

1. Write a class named `Account` to model accounts. The UML diagram for the class is shown in Figure 4.0. Write a test program to test the `Account` class. In the client program, create an `Account` object with an account ID of 1222, a balance of 20000, and an annual interest rate of 4.5%. Use the withdraw method to withdraw $2500, use the deposit method to deposit $3000, and print the balance and the monthly interest.

Account
-id:int -balance:double -annualInterestRate:double
+Account() +getId():int +getBalance():double +getAnnualInterestRate:double +setId(id:int):void +setBalance(bal:double):void +setAnnualInterestRate(rate:double):void +getMonthlyInterestRate():double +withdraw(amount:double):void +deposit(amount:double):void

Figure 4.0

Exercise:

1. Write a class named `Rectangle` to represent rectangles. The UML diagram for the class is shown in Figure 4.1 Suppose that all the rectangles are the same colour. Use a static variable for colour.

Rectangle
-width:double -height:double -color:String
+Rectangle() +Rectangle(width:double, height:double, color:String) +getWidth():double +setWidth(width:double):void +getHeight():double +setHeight(height:double):void +getColor:String +setColor(color:String):void +findArea():double +findPerimeter():double

Figure 4.1

Write a client program to test the class `Rectangle`. In the client program, create two `Rectangle` objects. Assign width 5 and height 50 each of the objects. Assign colour yellow. Display the properties of both objects and their areas.

2. Write a class named `Fan` to model fans. The properties, as shown in Figure 4.2, are speed, on, radius, and color. You need to provide the accessor and mutator methods for the properties, and the `toString` method for returning a string consisting of all the values of all the properties in this class. Suppose the fan has three fixed speeds. Use constants 1, 2, and 3 to denote slow, medium, and fast speed.

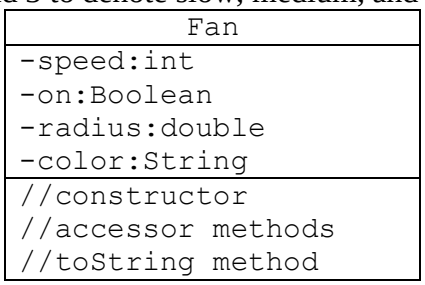

Figure 4.2

   Write a client program to test the `Fan` class. In the client program, create a `Fan` object. Assign maximum speed, radius 10, color yellow, and turn it on. Display the object by invoking its `toString` object.

3. Java API has the `GregorianCalendar` class in the `java.util` package that can be used to obtain the year, month and day of a date. The no-arg constructor constructs an instance for the current date and the methods `get(GregorianCalendar.YEAR)`, `get(GregorianCalendar.MONTH)`, and `get(GregorianCalendar.DAY)` return the year, month and day. Write a program to test this class to display the current year, month and day.

4. Write a class called `Time`. The `Time` class contains data fields hour, minute and second with their respective get methods. The no-arg constructor sets the hour, minute, and second for the current time in GMT. The current time can be obtained using `System.currentTime()`. Write a client program to test the `Time` class. In the client program, create a `Time` object and display hour, minute and second using the get methods.

5. Using the `Time` class above, create an array storing `Time` object with its associated data (hour, minute, and second). `Time` object is created for every 5 seconds. Display the Time using `toString` method. The `toString` method returns hour:minute:second e.g., 1:30:30.

6. Consider the UML diagram below:

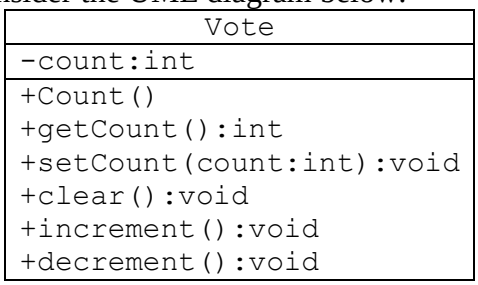

Figure 4.3

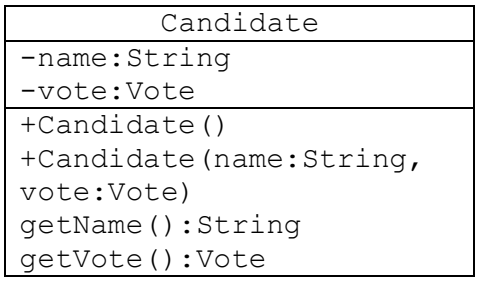

Figure 4.4

Develop a program that counts votes for two candidates for student body president. The input of vote is as follows:

Input	**Vote**
1	Increment Candidate1
2	Increment Candidate2
-1	Decrement Candidate1
-2	Decrement Candidate2
0	End the vote