**Lab #1**

**Applying OO Concepts Using Java**

|  |  |
| --- | --- |
| **Program 1** | Create a class called **Circle.java** with fields named radius, area, and diameter.  Include methods named setRadius(), getRadius, and computeDiameter(), which calculates the circle's diameter, and computeArea(), which computes the circle's area.  Hint:  diameter of a circle is twice its radius.   Hint:  area of a circle is 3.14 multiplied by the square of the radius.  Create a class called **TestCircle. java** whose main method declares 2 Circle objects.  Using setRadius() method, assign one Circle a small radius value and assign another a larger radius value.  Call computeDiameter() and computeArea for each circle and display the results.  See [solution](https://www2.bc.edu/brian-bernier/MC697/Labs/Solutions/Lab6/CircleExample.txt) for this lab. |

|  |  |
| --- | --- |
| **Program 2** | Create a class named **Student**.  A student has fields for an ID number, number of credit hours earned, and number of points earned.  (For example, many schools compute grade point averages based on a scale of 4, so a three-credit-hour class in which a student earns an A is worth 12 points.  Also, create a class variable called lastStudentID which holds the last student ID number assigned to a student.  Initialize it to 0.  Each new student is assigned a student ID by adding one to this number.  Update lastStudentID each time a student is created.  Next, include methods to assign values to all fields separately.  For example, one method should assign ID number, another should assign credit hours earned, etc.  A Student also has a field for grade point average.  Include a method to compute the grade point average field by divinding points by credit hours earned.  Write methods to display the values in each Student field.  Write a class named **ShowStudent** that instantiates a Student object from the class you created above.  Set the necessary values and compute the Student grade point average, and then display all the values associated with the Student.  Create three different students with different values  See the solution to this problem:   [Student.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework6/Student.java) and [ShowStudent.java](https://www2.bc.edu/brian-bernier/MC697/Homework/Solutions/Homework6/ShowStudent.java) |

|  |  |
| --- | --- |
| **Program3** | Using the classes you created in above, create a constructor method for the Student class you created.  The constructor should initialize each Student's ID number by adding one to the lastStudentID variable.  The constructor should also initialize the grade point average to 4.0.  Write a program called **ShowStudent2.java** that shows the constructor works by instantiating an object and displaying the intial values. |

 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.

1. 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.

|  |
| --- |
| Fan |
| -speed:int  -on:Boolean  -radius:double  -color:String |
| //constructor  //accessor methods  //toString method |
| 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.

1. 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.
2. 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.
3. 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.
4. Consider the UML diagram below:

|  |
| --- |
| Vote |
| -count:int |
| +Count()  +getCount():int  +setCount(count:int):void  +clear():void  +increment():void  +decrement():void |
| Figure 4.3 |

|  |
| --- |
| Candidate |
| -name:String  -vote:Vote |
| +Candidate()  +Candidate(name:String, vote:Vote)  getName():String  getVote():Vote |
| 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 |