ProblemSet number7 = new ProblemSet(“Classes and Objects”);

Problem Set 7 – Classes and Objects

This assignment must be completed and submitted before 11:59pm on Friday 12/9/2016.

**Part A: Reading and Review Questions (20 points total)**

Read Chapter 6.1-6.5 (pages 185 to 212) of Fundamentals of Java and answer the review questions below in a Microsoft Word document and save as:

**partA.yourFirstNameLastName.doc**

1. Describe the difference between object-oriented programming and procedural programming.

OOP groups methods and data together in classes/objects.

1. What is an object? How is an object different from a class.  
   A class is a blueprint or pattern used to create an object. It reserves no memory space for data. A class can be thought of as the type of an object variable. Each object has it’s own data space and is an instance of a class.
2. What is the difference between an accessor and a mutator? What naming conventions are used with accessors and mutators? The naming conventions it to prefix the corresponding instance variable with "get" or “set”.   
   An accessor (aka getter) gets/returns the value of an object’s member variable. Mutators (aka setters) are methods to get/change the values of instances variables.
3. What is a constructor? How is a constructor different from other methods?  
   Constructors are similar to methods and are used to initialize an object. Constructor look like methods except that they use the name of the class and have no return type.
4. Both methods and variables in a class are declared as either private or public. Describe the difference between private and public and indicate how a programmer decides which parts of a class should be private and which public.   
   Anything public can be directly referenced from outside the object. Private cannot be referenced externally but can be used anywhere inside the class definition.

**Part B: Simple Exercises (20 points total)**

Solve each of these problems first without a computer (then check your answer with a computer). If your answers do not match, try again until you know why the computer gave the output. Type up answers to the following problems in a Microsoft Word document and save as:

**partB.yourFirstNameLastName.doc**

Problem 1: (5 points)

What are two major problems with the following constructor?  
The “void” keyword is not used for constructors since they do not return a value. The integer variables x and y should be declared within the class outside of the constructor. As defined below neither x or y will be accessible anywhere else within the class.

public void Point (int initialX, int initialY)

{

int x = initialX;

int y = initialY;

}

Problem 2: (9 points)

Consider a class that represents a bank account.

1. Such a class might store information about the account balance, the name of the account holder, and an account number. What instance variables would you declare to hold this information? Give a type and name for each.   
    private double acctBal=0.0d;  
    private String acctName="";  
    private int acctNum=0;
2. There are a number of operations that would make sense for a bank account—withdraw money, deposit money, check the balance, and so on. Write a method header with return type, name, and parameter list, for each such operation described below. Don't write the whole method—just the header. They will all be public methods.
3. Withdraw a given amount from the account. This changes the account balance, but does not return a value.   
    public void withdraw (double w) {  
    …  
    }
4. Deposit a given amount into the account. This changes the account balance, but does   
   not return a value.   
    public void deposit (double d) {  
    …   
    }
5. Get the balance from the account. This does not change anything in the account; it simply returns the balance.

public double getBal () {

…

}

1. Return a string containing the account information (name, account number, balance). This does not change anything in the account.   
    public String acctInfo () {  
    return "Name="+acctName+" Acct#="+acctNum+" balance="+acctBal;  
   }
2. Charge a $10 fee. This changes the account balance but does not return a value.  
    public void fee (double fee) {  
    …  
    }
3. Create a new account given an initial balance, the name of the owner, and the account number. Note that this will be a constructor, and that a constructor does not have a return type.   
   BankAccount a = new BankAccount (1000000d,"Tank, Mr. Shark",666);

Problem 3: (6 points)

The following class represents a moving ball on a rectangular pool table.

public class MovingBall  
{  
     private int mLength, mWidth;  
     private int mPosX, mPosY;  
     private int mDirX, mDirY;

     public MovingBall(int length, int width, int dx, int dy)  
     {  
          mLength = length;  
          mWidth = width;  
          mPosX = length / 2;  
          mPosY = width / 2;  
          mDirX = dx;  
          mDirY = dy;  
     }

     public void move()  
     {  
          mPosX += mDirX;  
          mPosY += mDirY;  
          if (mPosX == 0 || mPosX == mLength) mDirX = -mDirX;  
          if (mPosY == 0 || mPosY == mWidth) mDirY = -mDirY;  
     }  
}

Given:

MovingBall b = new MovingBall(8, 4, 1, -1);

What are the values of mPosX and mPosY after 70 moves (i.e., 70 calls to b.move())?

Move() causes points to trace around this figure 8 type shape (see below). It takes 16x to go each trip around and be back at the origin/beginning point (4,2). 64x around your back at the origin/beginning plus 6x more (or total 70) moves you to the top of the right side or (6,4).

/\ (y axis)

5 |

4 | o **o**

3 | o o o o

2 o o o

1 | o o o o

0 -----|----o---|-----|----|----o----|----|----------🡪 (x axis)

| 1 2 3 4 5 6 7 8

|

|

V

**Part C: Programming Problems (60 points total)**

Problem 1: A Bank Account Class (20 points) Use file **Account.java** and **ManageAccounts.java**

1. File **Account.java** contains a partial definition for a class representing a bank account. Study the partial definition and complete the Account class as described below. Note that you won't be able to test your methods until you write **ManageAccounts** in part 2.
   1. Fill in the code for method deposit, which should add the deposited amount to the balance for the account
   2. Fill in the code for method toString, which should return a string containing the name, account number, and balance for the account.
   3. Fill in the code for method chargeFee, which should deduct a service fee from the account and return the new balance.
   4. Fill in the code for method changeName which takes a string as a parameter and changes the name on the account to be that string.
2. File **ManageAccounts.java** contains a program that uses the Account class above.
   1. Complete it as indicated by the comments.
   2. Modify ManageAccounts so that it prints the balance after the calls to chargeFees. Instead of using the getBalance method like you did after the deposit and withdrawal, use the balance that is returned from the chargeFees method. You can either store it in a variable and then print the value of the variable, or embed the method call in a println statement.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Account.java

//

// A bank account class with methods to deposit to, withdraw from,

// change the name on, charge a fee to, and print a summary of the account.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public class Account

{

private double balance;

private String name;

private int acctNum;

//Constructor -- initializes balance, owner, and account number

public Account(double initBal, String owner, int number)

{

balance = initBal;

name = owner;

acctNum = number;

}

// Checks to see if balance is sufficient for withdrawal.

// If so, decrements balance by amount; if not, prints message.

public void withdraw(double amount)

{

if (balance >= amount)

balance -= amount;

else

System.out.println("Insufficient funds");

}

// Adds deposit amount to balance.

public void deposit(double amount)

{

}

// Returns balance.

public double getBalance()

{

return balance;

}

// Returns a string containing the name, account number, and balance.

public String toString()

{

}

// Deducts $10 service fee

public double chargeFee()

{

}

// Changes the name on the account

public void changeName(String newName)

{

}

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ManageAccounts.java

//

// Use Account class to create and manage Sally and Joe's

// bank accounts

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

public class ManageAccounts

{

public static void main(String[] args)

{

Account acct1, acct2;

//create account1 for Sally with $1000

acct1 = new Account(1000, "Sally", 1111);

//create account2 for Joe with $500

//deposit $100 to Joe's account

//print Joe's new balance (use getBalance())

//withdraw $50 from Sally's account

//print Sally's new balance (use getBalance())

//charge fees to both accounts

//change the name on Joe's account to Joseph

//print summary for both accounts

}

}

Problem 2: (20 points) Use file **Name.java** and **NameTest.java**

Create a class called Name that represents a person’s name.

* The class should have fields representing the person’s first name, middle initial, and last name, named firstName, lastName, and middleInitial
* Encapsulate the Name class. Make its fields private and add appropriate accessor methods (getFirstName, getMiddleInitial, getLastName) to the class.
* Add mutator methods called setFirstName, setMiddleInitial, and setLastName to your Name class. Give the parameters the same names as your fields and use the this keyword in your solution.
* Add a constructor to the Name class that accepts a first name, middle initial, and last name as parameters and initializes the Name object’s state with those values.
* Add the five methods below:
  + public String getNormalOrder()
    - This method should return the person’s name in normal order, with the first name followed by the middle initial and last name. For example, if the first name is “John”, the middle initial is “Q”, and the last name “Public”, returns “John Q. Public”.
  + public String getReverseOrder()
    - Returns the person’s name in reverse order, with the last name preceding the first name and middle initial. For example if the first name is “John”, the middle initial is “Q”, and the last name “Public”, returns “Public, John Q.”
  + public boolean equals(Name otherName)
    - returns true if this name is the same as otherName. Comparisons should not be case sensitive.
  + public String initials()
    - returns the person's initials (a 3-character string). The initials should be all in upper case, regardless of what case the name was entered in.
  + public int length()
    - returns the total number of characters in the full name, not including spaces.

Now write a program NameTest.java creates two Name objects (a scanner is not required). You'll need first, middle, and last to creates a Name object for each, and uses the methods of the Name class to do the following:

1. For each name, print
   * first-middle-last version
   * last-first-middle version
   * initials
   * length
2. Tell whether or not the names are the same

Problem 3: (20 points) Use file **Pizza.java**

Pizzas can be ordered in several sizes. Some pizza parlors cut all pizzas into 8 slices, while others get more slices out of the larger pizza pies. It might be useful to know what size pie offers the best deal in terms of cost per slice or cost per square inch of pizza. Here’s a Java program to help determine these facts:

public class PizzaTester

{

public static void main (String args [] )

{

Pizza p = new Pizza();

p.inputTheValues();

System.out.println( "That's $" + p.costPerSlice() +

" per slice." );

System.out.println(" which is $" + p.costPerSquareInch()

+ " per square inch!");

}

}

When the Pizza class is properly defined, here is what your program will look like in action:

Enter the radius of the pizza, in inches: **8**

Enter the price of the pizza: $**10.50**

Enter the number of slices in each pizza: **10**

That's $1.05 per slice, which is $0.05222271570202816 per square inch!

Enter the radius of the pizza, in inches: **10**

Enter the price of the pizza: $**11.95**

Enter the number of slices in each pizza: **8**

That's $1.49375 per slice, which is $0.03803803139896298 per square inch!

Your job is to define **Pizza.java**. Use the PizzaTester class above to test your Pizza class. The Pizza class must contain 3 instance variables and 3 instance methods (named inputTheValues(), costPerSlice(), and costPerSquareInch()).

By the way, the area of a circle is “PI” times the radius squared. The constant “PI” is equal to 3.14159265358979 3, but you should simply use Math.PI in your program.

Extra Credit: (10 points) Use file **Employee.java and EmployeeTester.java**

Create an employee class with the following instance variables:

String name;

int age;

String jobTitle;

double salary;

Create appropriate constructors, mutators, accessors, one talkAboutSelf() method, and at least two additional useful clever methods of your choice (for example: fireEmployee, giveRaise, etc. be clever!!!). Include these in a self-created tester method that demonstrates the full effects of your Employee class.

//Example of Employee Tester Below:

Employee chio = new Employee (“Andrew Chiofolo”, 85, “math teacher”, “$50,000”);

System.out.println(chio.talkAboutSelf());

//Sample Output

Hi! My name is Andrew Chiofolo. I am an 85 year old math teacher who makes $50,000 per year.

**

*You are finished! Create a “.zip” of your work so that you can electronically submit it on Schoology. Click the link that says “Problem Set 7 Homework Upload.” and attach the zip file.*