**Inheritance related Problem Statements**

1. Write an inheritance hierarchy for classes Quadrilateral, Trapezoid, Parallelogram,

Rectangle and Square. Use Quadrilateral as the superclass of the hierarchy. Create and use a Point class to represent the points in each shape. Make the hierarchy as deep (i.e., as many levels) as possible. Specify the instance variables and methods for each class. The private instance variables of Quadrilateral should be the *x-y* coordinate pairs for the four endpoints of the Quadrilateral.

Write a program that instantiates objects of your classes and outputs each object’s area (except Quadrilateral)

1. Using the Account class as a base class, write two derived classes alled SavingsAccount and CurrentAccount. A SavingsAccount object, in addition to the attributes of an Account object, should have an interest variable and a method which adds interest to the account. A CurrentAccount object, in addition to the attributes of an Account object, should have an overdraft limit variable. Ensure that you have overridden methods of the Account class as necessary in both derived classes.

Now create a Bank class, an object of which contains an array of Account objects. Accounts in the array could be instances of the Account class, the SavingsAccount class, or the CurrentAccountclass. Create some test accounts (some of each type).

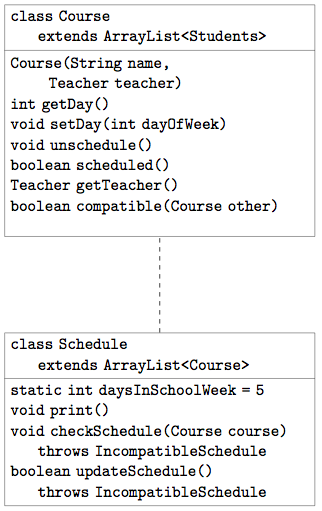
Write an update method in the bank class. It iterates through each account, updating it in the following ways: Savings accounts get interest added (via the method you already wrote); CurrentAccounts get a letter sent if they are in overdraft.

The Bank class requires methods for opening and closing accounts, and for paying a dividend into each account.

Hints:

* Note that the balance of an account may only be modified through the deposit(double) and withdraw(double) methods.
* The Account class should not need to be modified at all.
* Be sure to test what you have done after each step.

1. Course and Schedule hierarchy



Create a class Course. Each course has a single Teacher and many students.

* Think about how to add students to the course. What do you need to do that make that possible?
* Each course is either scheduled or not scheduled for a specific day of the week, which we can represent by an int.
* The setDay(int) method schedules a course for a given day of the week.
* The method compatible checks if two courses can co-exist at the same time (that is, they do not share any People in common).

Next, we are going to write a class Schedule that tries to find a compatible schedule for a list of courses.