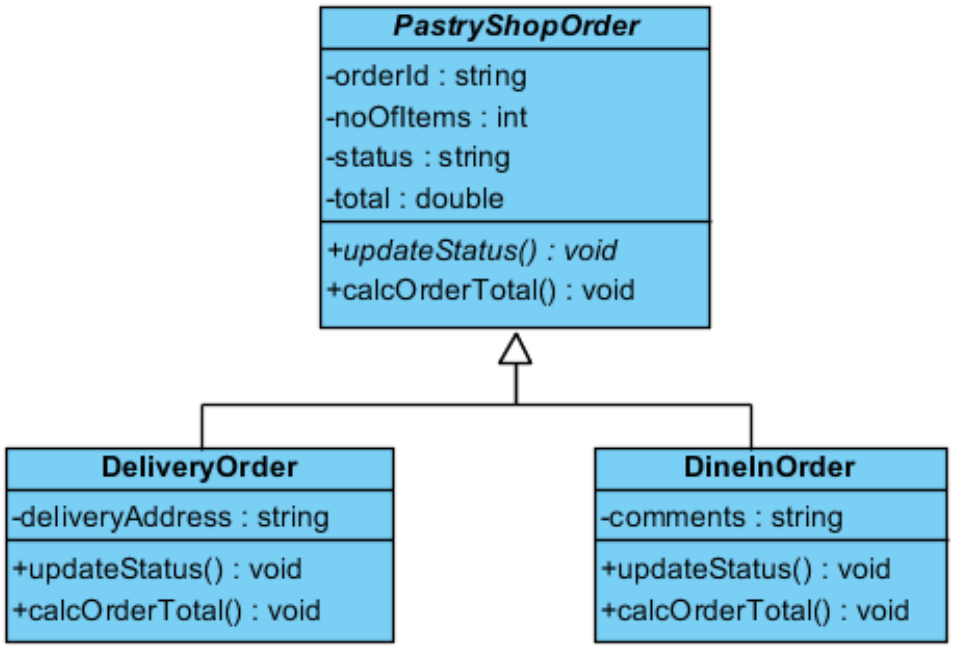
Object Oriented Development

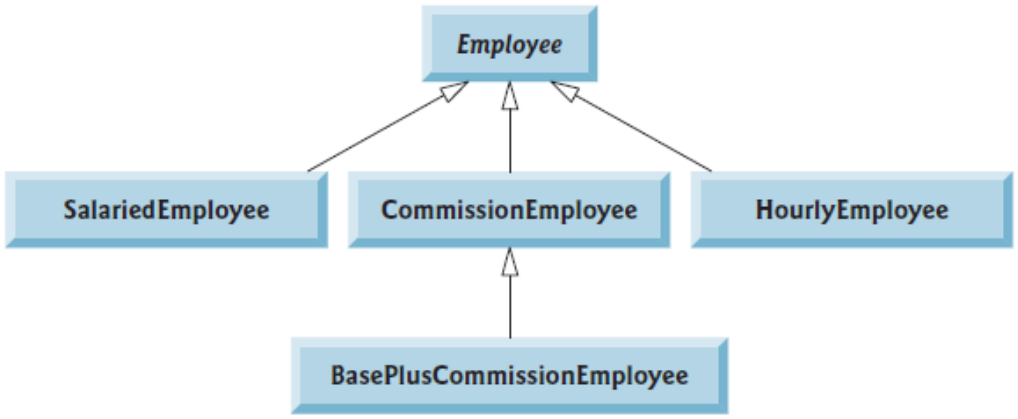
Week 5 – Tutorial 4

Q1 ***(15mins)***

Implement the below diagram using Java. Make sure to provide meaningful implementation for the behaviors.



Q2. ***(60 mins)***



Consider the Employee class hierarchy given above, identify suitable attributes for each class. The Employee must identify a private instance variable birthdate, use the Date class you created in Tutorial 3

(last week). Assume that payroll is processed once per month. Create an array of Employee variables to store references to the various employee objects. In a loop, calculate the payroll for each Employee (polymorphically), and add a $100.00 bonus to the person's payroll amount if the current month is the one in which the Employee's birthday occurs.

Modify the above payroll system to include an additional Employee subclass PieceWorker that represents an employee whose pay is based on the number of pieces of merchandise produced. Class PieceWorker should contain private instance variables wage (to store the employee's wage per piece) and pieces (to store the number of pieces produced). Provide a concrete implementation of method earnings in class PieceWorker that calculates the employee's earnings by multiplying the number of pieces produced by the wage per piece. Create an array of Employee variables to store references to objects of each concrete class in the new Employee hierarchy. For each Employee, display its String representation and earnings.

Q3. ***(45mins)***

The CarbonFootprint Interface: Using interfaces, (as you learned during the previous lecture,) you can specify similar behaviors for possibly disparate classes. Governments and companies worldwide are becoming increasingly concerned with carbon footprints (annual releases of carbon dioxide into the atmosphere) from buildings burning various types of fuels for heat, vehicles burning fuels for power, and the like. Many scientists blame these greenhouse gases for the phenomenon called global warming. Create three small classes unrelated by inheritance—classes Building, Car and Bicycle. Give each class some unique appropriate attributes and behaviors that it does not have in common with other classes. Write an interface CarbonFootprint with a getCarbonFootprint method. Have each of your classes implement that interface, so that its getCarbonFootprint method calculates an appropriate carbon footprint for that class (check out a few websites that explain how to calculate carbon footprints). Write an application that creates objects of each of the three classes and call the appropriate methods to calculate the carbon footprint.