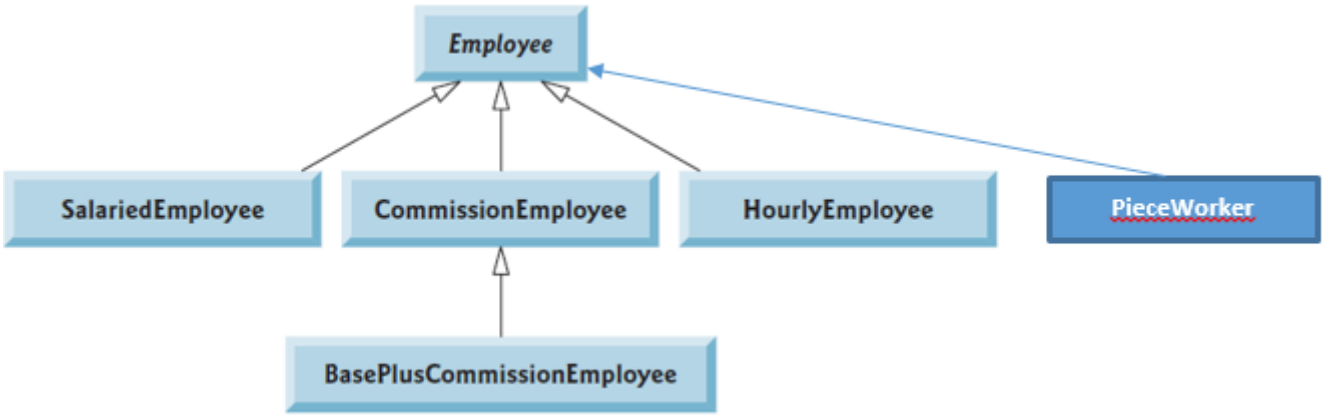
COMP212-Programming 03

Lab Assignment Exercise: Using Classes, Objects, Inheritance and Polymorphism

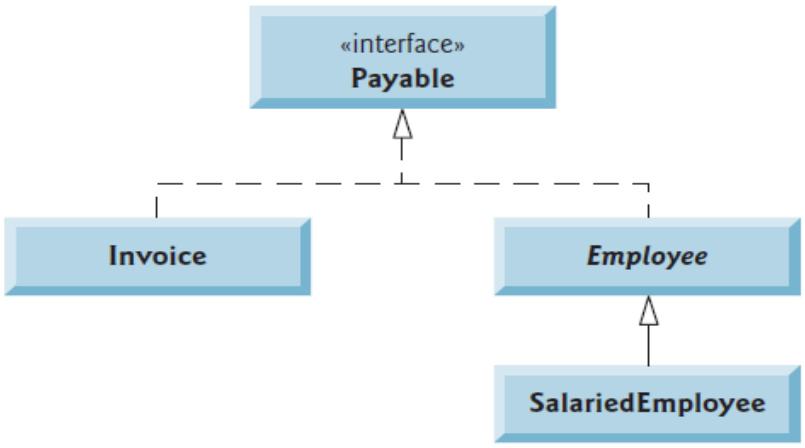
Marks/Weight: 30/2%

Exercise #1:
(Payroll System Modification)

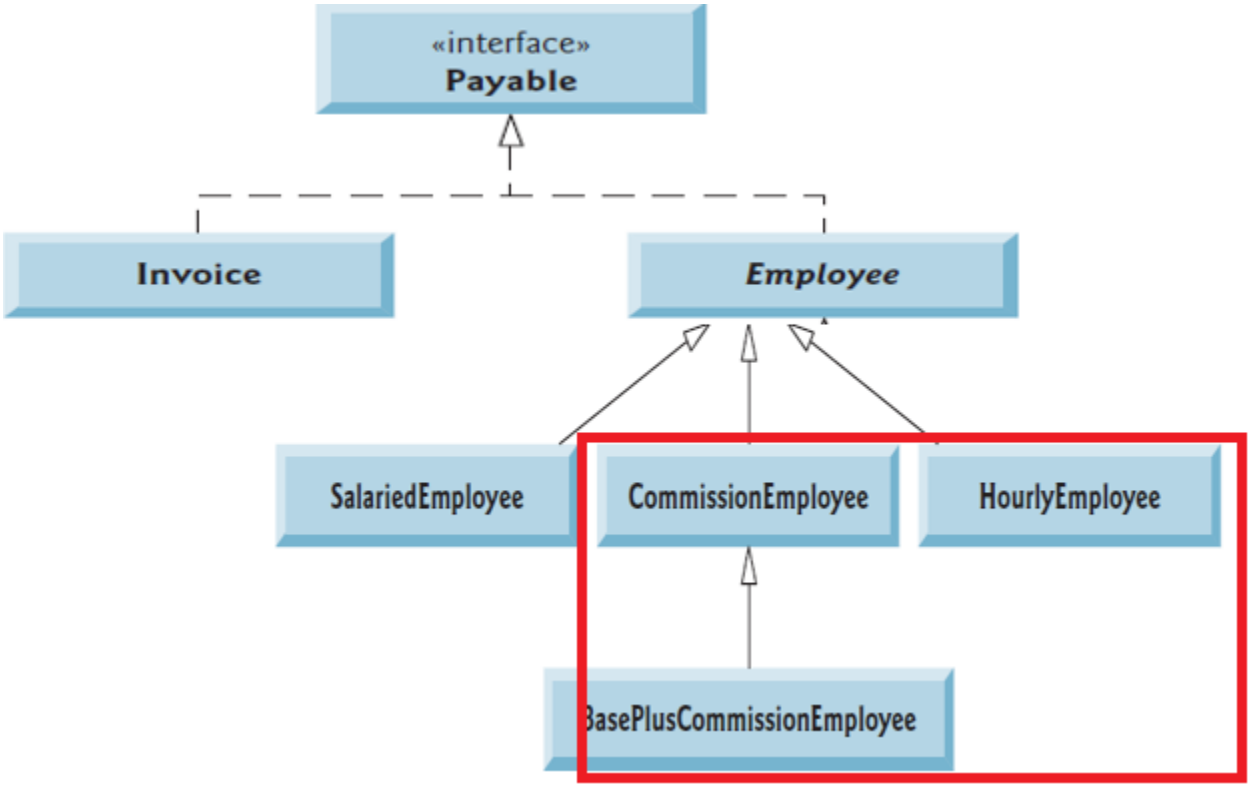


Modify the above payroll system which was implemented in the lab class, to include an additional Employee subclass **PieceWorker** that represents an employee whose pay is based on the number of pieces of merchandise produced.
Class PieceWorker should contain private instance variables wage (to store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

Exercise #2: (**Accounts Payable System Modification**)



[Previously implemented]

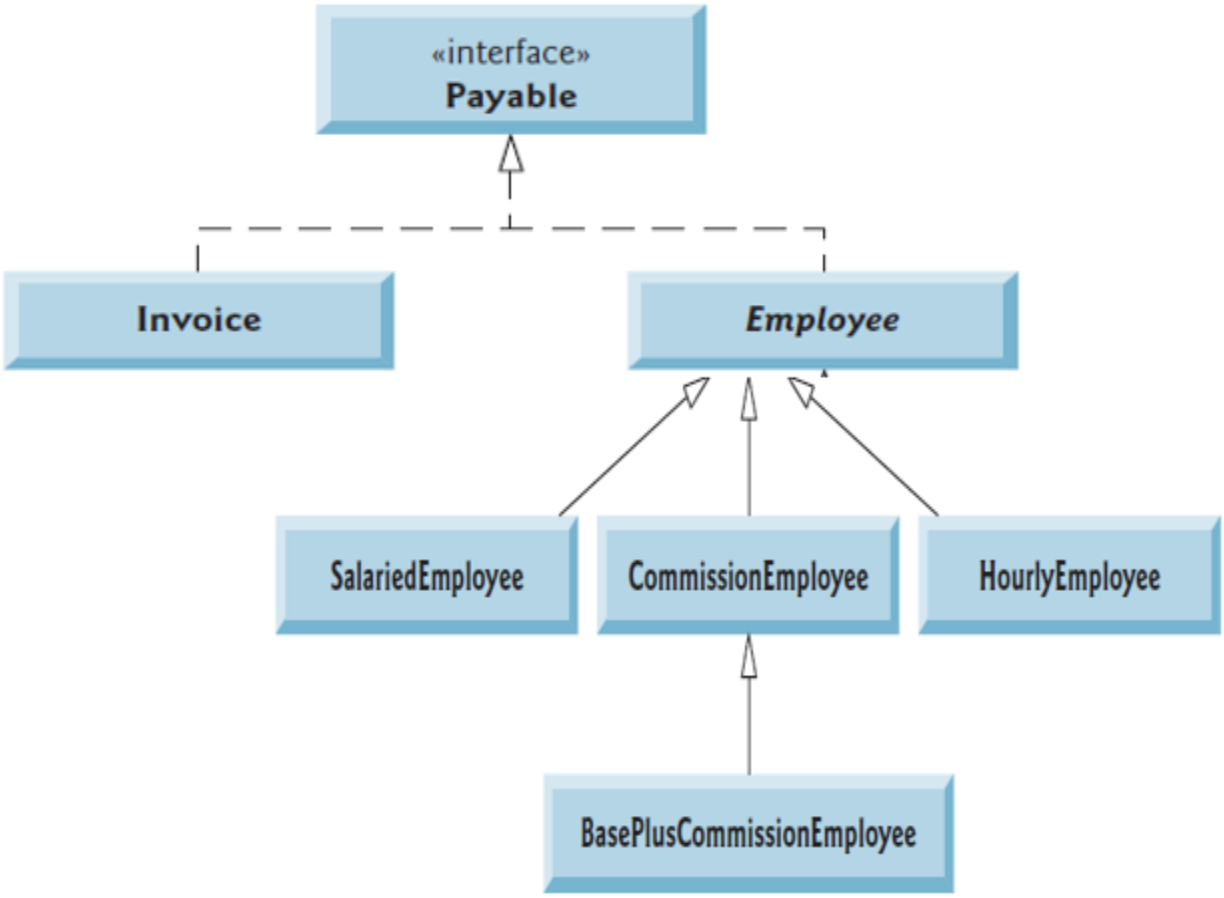


[Three classes to be added as shown in the box]

In this exercise, we modify the above accounts payable application (covered in the class) to include the complete functionality of the payroll application The application should still process two Invoice objects, but now should process one object of each of the four Employee subclasses.
If the object currently being processed is a Base-PlusCommissionEmployee, the application should increase the BasePlusCommissionEmployee's base salary by 10%. Finally, the application should output the payment amount for each object.
Complete the following steps to create the new application:
a) Modify classes HourlyEmployee and CommissionEmployee to place them in the Payable hierarchy as subclasses of the version of Employee that implements Payable. [Hint: Change the name of method earnings to getPaymentAmount in each subclass so that the class satisfies its inherited contract with interface Payable.]
b) Modify class BasePlusCommissionEmployee such that it extends the version of class CommissionEmployee created in part (a).

c) Modify PayableInterfaceTest to polymorphically process two Invoices, one SalariedEmployee, one HourlyEmployee, one CommissionEmployee and one Base-PlusCommissionEmployee.
First output a String representation of each Payable object.
Next, if an object is a BasePlusCommissionEmployee, increase its base salary by 10%.
If the object currently being processed is a HourlyEmployee, the application should increase the HourlyEmployee's hourly rate by 2.00 dollar.
Finally, the application should output the payment amount for each object.

Exercise #3:
(Accounts Payable System Modification)



[Three classes to be added without modifying the classes]

It's possible to include the functionality of the payroll application of Exercise 2.0 in the accounts payable application without modifying Employee subclasses SalariedEmployee, HourlyEmployee, CommissionEmployee or BasePlusCommission-Emplyee.
To do so, you can modify class Employee to implement interface Payable and declare method getPaymentAmount to invoke method earnings. Method getPaymentAmount would then be inherited by the subclasses in the Employee hierarchy.
When getPaymentAmount is called for a particular subclass object, it polymorphically invokes the appropriate earnings method for that subclass. Re-implement Exercise 2.0 using the original Employee hierarchy from the payroll application of Modify class Employee as described in this exercise, and do not modify any of class Employee's subclasses.