Lab Assignment #3

Due Date: Week 6 **Marks/Weightage:** 30/10%

Purpose: The purpose of this Lab assignment is to:

- Practice the use of Inheritance, Polymorphism and Exception Handling

References: Read the course's text "Java How to program, 11th edition Early Objects", **Chapters 9 to 11** and the lecture notes/ppts. This material provides the necessary information that you need to complete the exercises.

Instructions: Be sure to read the following general instructions carefully:

This lab should be completed individually by all the students. You will have to demonstrate your solution in a scheduled lab session and submitting the project **through drop box link on e-Centennial**.
You must start and name your Eclipse workspace according to the following rule:
FirstName_LastName _COMP228_ SectionNumber_Labnumber
For Example: **John_Smith _COMP228_Sec001_Lab03**

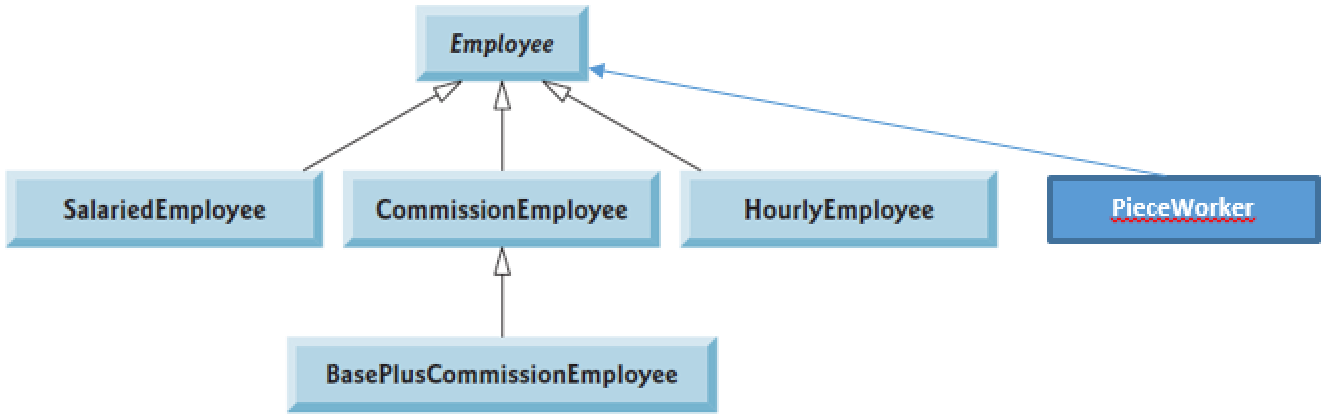
Each exercise should be placed in a separate project named *exercise1*, *exercise2*, etc.
You should have a package name as follows:
FirstName_LastName_Exercise01 and so on…
Submit your assignment in a **zip file** that is named according to the following rule:
FirstName_LastName _COMP228_ SectionNumber_Labnumber
For Example: **John_Smith _COMP228_Sec001_Lab03**

Apply the naming conventions for variables, methods, classes, and packages:
- *variable names* start with a *lowercase* character for the first word and uppercase for every other word
- *classes* start with an *uppercase* character of every word
- **packages** use only *lowercase* characters
- *methods* start with a *lowercase* character for the first word and uppercase for every other word

Note: You are required to be present during the in-class demonstration. Late submission will not be considered

Exercise #1: [10 marks]

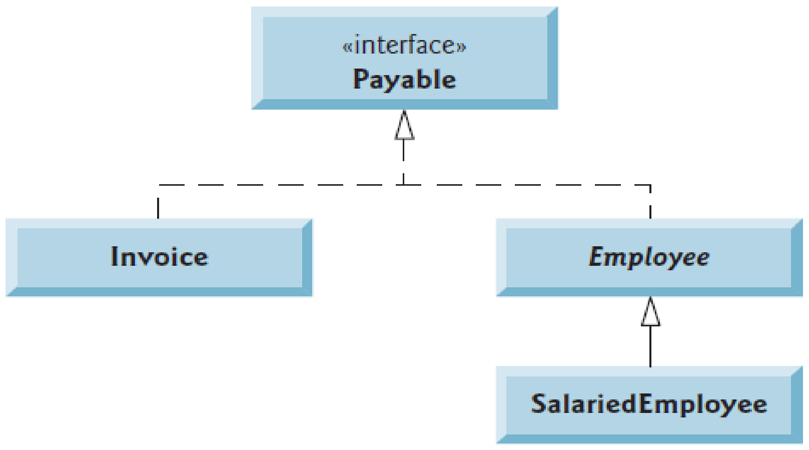
(Payroll System Modification)



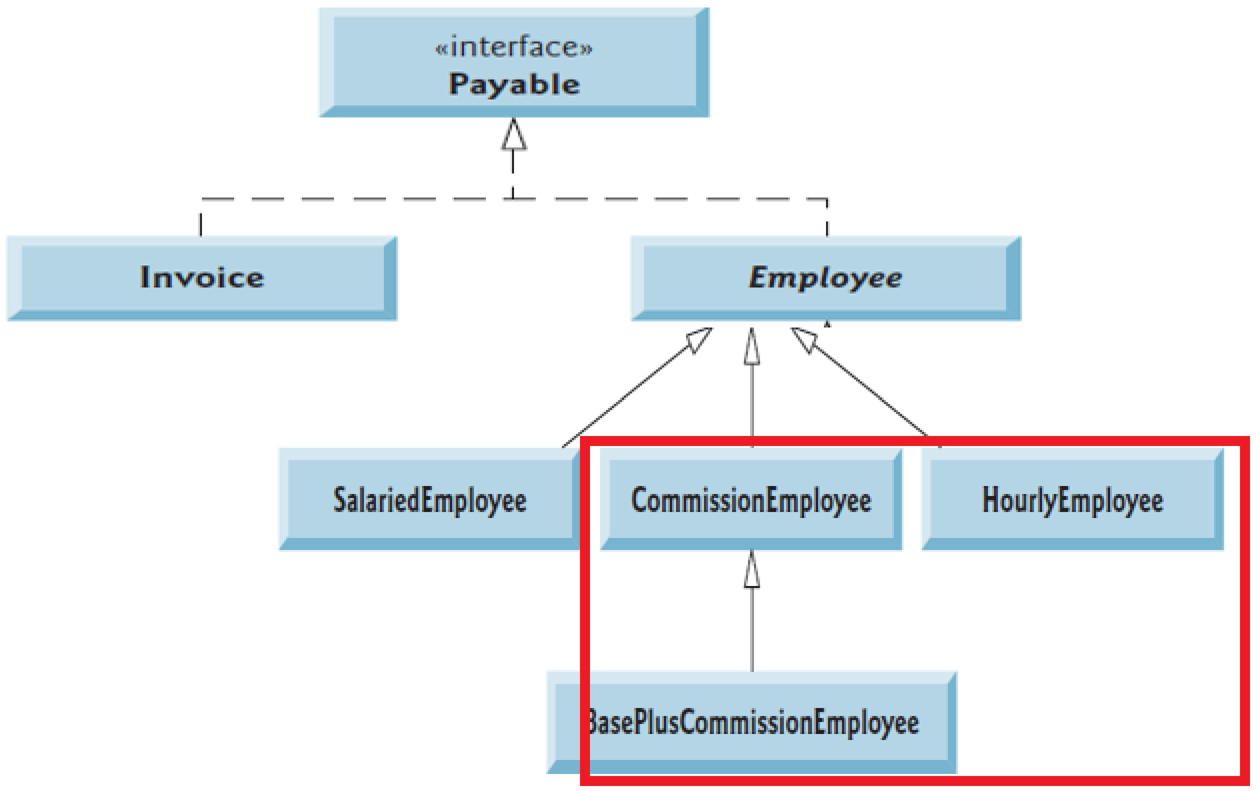
Modify the above payroll system which was implemented in the lab class, to include an additional Employee subclass **PieceWorker** that represents an employee whose pay is based on the number of pieces of merchandise produced.
Class PieceWorker should contain private instance variables wage (to store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

Exercise #2: [10 marks]

(**Accounts Payable System Modification**)



[Previously implemented]



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

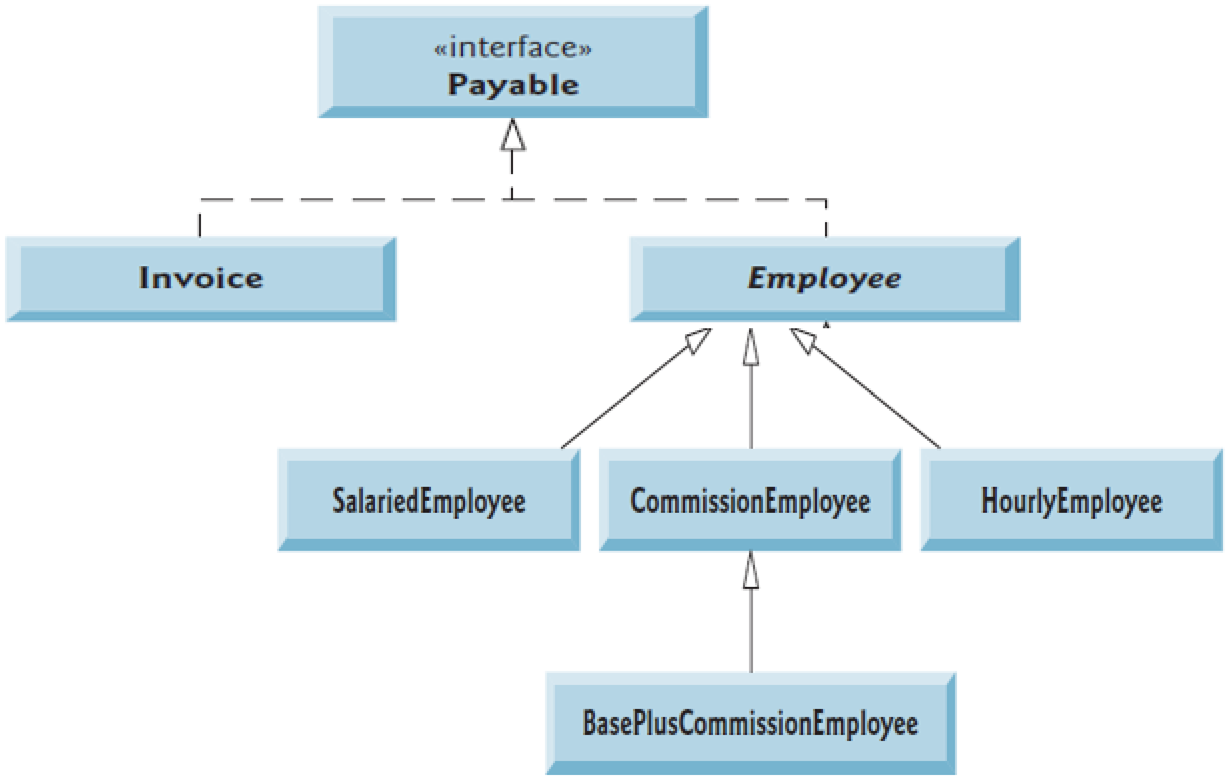
In this exercise, we modify the above accounts payable
application (covered in the class) to include the complete functionality of the payroll application
The application should still process two Invoice objects, but now should process one object of each of the four Employee subclasses.
If the object 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) **[10 marks]**



[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.