CSULB
CECS 277

Lab4 (comparable)

Use an abstract method and polymorphism to perform payroll calculations based on an employee inheritance hierarchy that meets the following requirements:

A company pays its employees on a weekly basis. The employees are of four types:

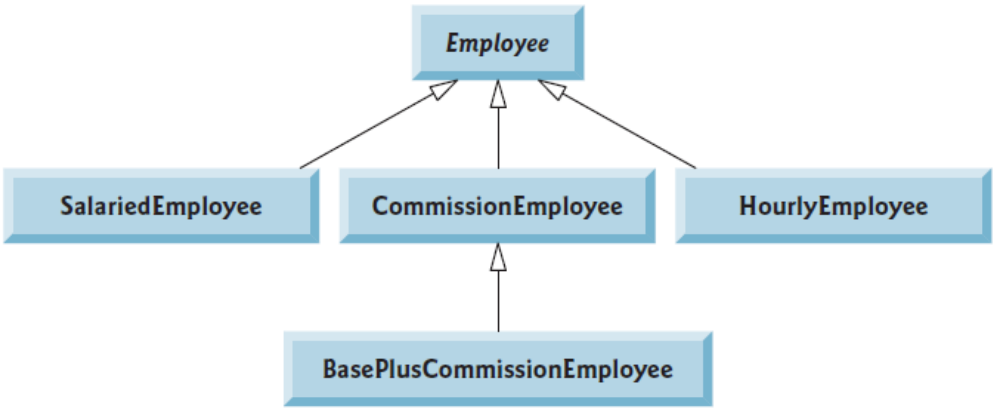
1. **Salaried employees are paid a fixed weekly salary regardless of the number of hours worked,**
2. **Hourly employees are paid by the hour and receive overtime pay (i.e., 1.5 times their hourly salary rate) for all hours worked in excess of 40 hours,**
3. **Commission employees are paid a percentage of their sales**
4. **Base-salaried commission employees receive a base salary plus a percentage of their sales. For the current pay period, the company has decided to reward salaried-commission employees by adding 10% to their base salaries.**

The company wants us to write an application that performs its payroll calculations polymorphically.

Use **abstract class Employee** to represent the general concept of an employee.

The classes that extend Employee are SalariedEmployee, CommissionEmployee and HourlyEmployee. Class BasePlusCommissionEmployee—which extends CommissionEmployee— represents the last employee type.

The UML class diagram shows the inheritance hierarchy for our polymorphic employee-payroll application. Abstract class name Employee is italicized—a convention of the UML.



Now complete and use the following tester

```
// PayrollSystemTest.java
// Employee hierarchy test program.

public class PayrollSystemTest
 {
 public static void main( String[] args )
 {
   // create subclass objects
   SalariedEmployee salariedEmployee = new SalariedEmployee( "John", "Smith", "111-11-
1111", 800.00 );
   HourlyEmployee hourlyEmployee = new HourlyEmployee( "Karen", "Price", "222-22-2222",
16.75, 40 );
  CommissionEmployee commissionEmployee = new CommissionEmployee("Sue", "Jones",
"333-33-3333", 10000, .06 );
  BasePlusCommissionEmployee basePlusCommissionEmployee = new
BasePlusCommissionEmployee
                        ( "Bob", "Lewis", "444-44-4444", 5000, .04, 300 );

  System.out.println( "Employees processed individually:\n" );
  System.out.printf( "%s\n%s: $%,.2f\n\n",salariedEmployee, "earned",
salariedEmployee.earnings() );

  System.out.printf( "%s\n%s: $%,.2f\n\n", hourlyEmployee, "earned",
hourlyEmployee.earnings() );
  System.out.printf( "%s\n%s: $%,.2f\n\n", commissionEmployee, "earned",
commissionEmployee.earnings() );
  System.out.printf( "%s\n%s: $%,.2f\n\n", basePlusCommissionEmployee,  "earned",
basePlusCommissionEmployee.earnings() );


Now // create four-element Employee array

 Now /// initialize array with Employees objects that you declared above


// generically process each element in array employees
for ( Employee currentEmployee : employees )
  {
     // invokes toString    to output the current employee

     // if the element is a BasePlusCommissionEmployee
           // downcast Employee reference to BasePlusCommissionEmployee reference calculate
           // and output the  new base salary with 10%% increase is
//else, just print the currentEmployee earnings

} //end for

// get type name of each object in employees array
for ( int j = 0; j < employees.length; j++ )
    System.out.printf( "Employee %d is a %s\n", j, employees[ j ].getClass().getName() );
} // End  main.
```

Have your Employee class implements comparable interface (compare employees based on the second letter of their first name.

For each class, you should define toString method, equals and hashCode methods.

Draw the class UML.

For today the minimum requirement is to submit a code representing the implementation of **compareTo** method