Lab Exercises

Name:

Lab Exercise 2 — Accounts Payable System Modification

Name: _____ **Date:** _____

Section: _____

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 10.6–Fig. L 10.9)
5. Problem-Solving Tips
6. Follow-Up Question and Activity

The program template represents a complete working Java program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with Java code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up question. The source code for the template is available at `www.pearsonhighered.com/deitel`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 10 of *Java How to Program: 8/e*. In this lab you will practice:

- Provide additional polymorphic processing capabilities to an inheritance hierarchy by implementing an interface.
- Using the `instanceof` operator to determine whether a variable refers to an object that has an *is-a* relationship with a particular class.

The follow-up question and activity will also give you practice:

- Comparing `interfaces` and `abstract` classes.

Description of the Problem

(Accounts Payable System Modification) In this exercise, we modify the accounts payable application of Figs. 10.11–10.15 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 (Figs. 10.5–10.8). If the object currently being processed is a `BasePlusCommissionEmployee`, 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` (Fig. 10.13). [*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*.

Lab Exercises

Name:

Lab Exercise 2 — Accounts Payable System Modification

c) Modify `PayableInterfaceTest` to polymorphically process two `Invoices`, one `SalariedEmployee`, one `HourlyEmployee`, one `CommissionEmployee` and one `BasePlusCommissionEmployee`. First output a string representation of each `Payable` object. Next, if an object is a `BasePlusCommissionEmployee`, increase its base salary by 10%. Finally, output the payment amount for each `Payable` object.

Sample Output

```
Invoices and Employees processed polymorphically:

invoice:
part number: 01234 (seat)
quantity: 2
price per item: $375.00
payment due: $750.00

invoice:
part number: 56789 (tire)
quantity: 4
price per item: $79.95
payment due: $319.80

salaried employee: John Smith
social security number: 111-11-1111
weekly salary: $800.00
payment due: $800.00

hourly employee: Karen Price
social security number: 222-22-2222
hourly wage: $16.75; hours worked: 40.00
payment due: $670.00

commission employee: Sue Jones
social security number: 333-33-3333
gross sales: $10,000.00; commission rate: 0.06
payment due: $600.00

base-salaried commission employee: Bob Lewis
social security number: 444-44-4444
gross sales: $5,000.00; commission rate: 0.04; base salary: $300.00
new base salary with 10% increase is: $330.00
payment due: $530.00
```

Program Template

```
1  // Lab Exercise 2: HourlyEmployee.java
2  // HourlyEmployee class extends Employee, which implements Payable.
3
4  public class HourlyEmployee extends Employee
5  {
6     private double wage; // wage per hour
7     private double hours; // hours worked for week
8
```

Fig. L 10.6 | `HourlyEmployee.java`. (Part 1 of 2.)

Lab Exercises

Name:

Lab Exercise 2 — Accounts Payable System Modification

```
 9     // five-argument constructor
10     public HourlyEmployee( String first, String last, String ssn,
11        double hourlyWage, double hoursWorked )
12     {
13        super( first, last, ssn );
14        setWage( hourlyWage ); // validate and store hourly wage
15        setHours( hoursWorked ); // validate and store hours worked
16     } // end five-argument HourlyEmployee constructor
17
18     // set wage
19     public void setWage( double hourlyWage )
20     {
21        wage = ( hourlyWage < 0.0 ) ? 0.0 : hourlyWage;
22     } // end method setWage
23
24     // return wage
25     public double getWage()
26     {
27        return wage;
28     } // end method getWage
29
30     // set hours worked
31     public void setHours( double hoursWorked )
32     {
33        hours = ( ( hoursWorked >= 0.0 ) && ( hoursWorked <= 168.0 ) ) ?
34           hoursWorked : 0.0;
35     } // end method setHours
36
37     // return hours worked
38     public double getHours()
39     {
40        return hours;
41     } // end method getHours
42
43     // calculate earnings; implement interface Payable method not
44     // implemented by superclass Employee
45     /* write a method header to satisfy the Payable interface */
46     {
47        if ( getHours() <= 40 ) // no overtime
48           return getWage() * getHours();
49        else
50           return 40 * getWage() + ( getHours() - 40 ) * getWage() * 1.5;
51     } // end method getPaymentAmount
52
53     // return String representation of HourlyEmployee object
54     public String toString()
55     {
56        return String.format( "hourly employee: %s\n%s: $%,.2f; %s: %,.2f",
57           super.toString(), "hourly wage", getWage(),
58           "hours worked", getHours() );
59     } // end method toString
60  } // end class HourlyEmployee
```

Fig. L 10.6 | `HourlyEmployee.java`. (Part 2 of 2.)

Lab Exercises

Name:

Lab Exercise 2 — Accounts Payable System Modification

```
// Lab Exercise 2: CommissionEmployee.java
// CommissionEmployee class extends Employee, which implements Payable.

public class CommissionEmployee extends Employee
{
   private double grossSales; // gross weekly sales
   private double commissionRate; // commission percentage

   // five-argument constructor
   public CommissionEmployee( String first, String last, String ssn,
      double sales, double rate )
   {
      super( first, last, ssn );
      setGrossSales( sales );
      setCommissionRate( rate );
   } // end five-argument CommissionEmployee constructor

   // set commission rate
   public void setCommissionRate( double rate )
   {
      commissionRate = ( rate > 0.0 && rate < 1.0 ) ? rate : 0.0;
   } // end method setCommissionRate

   // return commission rate
   public double getCommissionRate()
   {
      return commissionRate;
   } // end method getCommissionRate

   // set gross sales amount
   public void setGrossSales( double sales )
   {
      grossSales = ( sales < 0.0 ) ? 0.0 : sales;
   } // end method setGrossSales

   // return gross sales amount
   public double getGrossSales()
   {
      return grossSales;
   } // end method getGrossSales

   // calculate earnings; implement interface Payable method not
   // implemented by superclass Employee
   /* write a method header to satisfy the Payable interface */
   {
      return getCommissionRate() * getGrossSales();
   } // end method getPaymentAmount

   // return String representation of CommissionEmployee object
   public String toString()
   {
      return String.format( "%s: %s\n%s: $%,.2f; %s: %.2f",
         "commission employee", super.toString(),
         "gross sales", getGrossSales(),
         "commission rate", getCommissionRate() );
   } // end method toString
} // end class CommissionEmployee
```

Fig. L 10.7 | CommissionEmployee.java.

Lab Exercises

Name:

Lab Exercise 2 — Accounts Payable System Modification

```
1  // Lab Exercise 2: BasePlusCommissionEmployee.java
2  // BasePlusCommissionEmployee class extends CommissionEmployee.
3
4  public class BasePlusCommissionEmployee extends CommissionEmployee
5  {
6     private double baseSalary; // base salary per week
7
8     // six-argument constructor
9     public BasePlusCommissionEmployee( String first, String last,
10       String ssn, double sales, double rate, double salary )
11    {
12       super( first, last, ssn, sales, rate );
13       setBaseSalary( salary ); // validate and store base salary
14    } // end six-argument BasePlusCommissionEmployee constructor
15
16    // set base salary
17    public void setBaseSalary( double salary )
18    {
19       baseSalary = ( salary < 0.0 ) ? 0.0 : salary; // non-negative
20    } // end method setBaseSalary
21
22    // return base salary
23    public double getBaseSalary()
24    {
25       return baseSalary;
26    } // end method getBaseSalary
27
28    // calculate earnings; override CommissionEmployee implementation of
29    // interface Payable method
30    /* write a method header to satisfy the Payable interface */
31    {
32       /* calculate and return the BasePlusCommissionEmployee's earnings */
33    } // end method getPaymentAmount
34
35    // return String representation of BasePlusCommissionEmployee object
36    public String toString()
37    {
38       return String.format( "%s %s; %s: $%,.2f",
39          "base-salaried", super.toString(),
40          "base salary", getBaseSalary() );
41    } // end method toString
42 } // end class BasePlusCommissionEmployee
```

Fig. L 10.8 | BasePlusCommissionEmployee.java.

```
1  // Lab Exercise 2: PayableInterfaceTest.java
2  // Tests interface Payable.
3
4  public class PayableInterfaceTest
5  {
6     public static void main( String args[] )
7     {
8        // create six-element Payable array
9        Payable payableObjects[] = new Payable[ 6 ];
10
```

Fig. L 10.9 | PayableInterfaceTest.java. (Part 1 of 2.)

Lab Exercises

Name:

Lab Exercise 2 — Accounts Payable System Modification

```
11         // populate array with objects that implement Payable
12         payableObjects[ 0 ] = new Invoice( "01234", "seat", 2, 375.00 );
13         payableObjects[ 1 ] = new Invoice( "56789", "tire", 4, 79.95 );
14         payableObjects[ 2 ] =
15            new SalariedEmployee( "John", "Smith", "111-11-1111", 800.00 );
16         payableObjects[ 3 ] =
17            /* create an HourlyEmployee object */
18         payableObjects[ 4 ] =
19            /* create a CommissionEmployee object */
20         payableObjects[ 5 ] =
21            /* create a BasePlusCommissionEmployee object */
22
23         System.out.println(
24            "Invoices and Employees processed polymorphically:\n" );
25
26         // generically process each element in array payableObjects
27         for ( Payable currentPayable : payableObjects )
28         {
29            // output currentPayable and its appropriate payment amount
30            System.out.printf( "%s \n", currentPayable.toString() );
31
32            /* write code to determine whether currentPayable is a
33               BasePlusCommissionEmployee object */
34            {
35               /* write code to give a raise */
36               /* write code to ouput results of the raise */
37            } // end if
38
39            System.out.printf( "%s: $%,.2f\n\n",
40               "payment due", currentPayable.getPaymentAmount() );
41         } // end for
42      } // end main
43   } // end class PayableInterfaceTest
```

Fig. L 10.9 | `PayableInterfaceTest.java`. (Part 2 of 2.)