# COP 3330 - Assignment 5

## Due: Tuesday April 1 at 11:59 pm.

Name: Andrew Downs\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: April 1, 2014\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Payroll System Modification**

The problem at hand is divided into six parts:

1. Assignment Objectives

2. Description of the Problem

3. Sample Output

4. Program Template (Fig. 10.4 and Fig. 10.5)

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 questions.

**Assignment 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:

• Creating a new class and adding it to an existing class hierarchy.

• Using the updated class hierarchy in a polymorphic application.

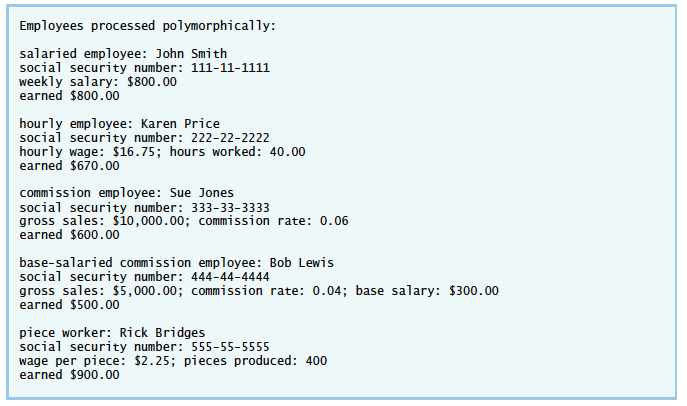
The follow-up question and activity also will give you practice:

• Understanding polymorphism.

**Description of the Problem**

(Payroll System Modification) Modify the payroll system of Figs. 10.4–10.9 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.

**Sample Output**



**Program Template**

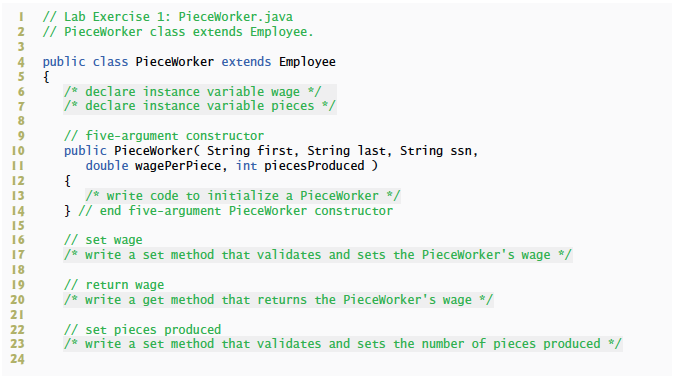


Fig. 10.4 | PieceWorker.java. (Part 1 of 2.)

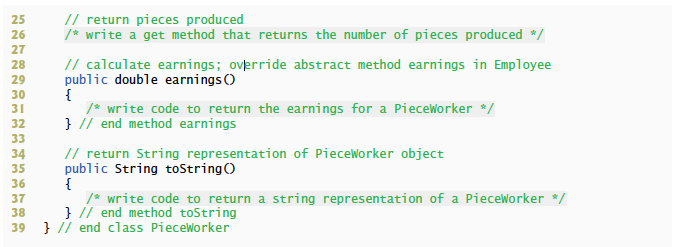


Fig. 10.4 | PieceWorker.java. (Part 2 of 2.)



Fig. L 10.5 | PayrollSystemTest.java

**Problem-Solving Tips**

1. The *PieceWorker* constructor should call the superclass *Employee* constructor to initialize the employee’s name.

2. The number of pieces produced should be greater than or equal to 0. Place this logic in the *set* method for the *pieces* variable.

3. The wage should be greater than or equal to 0. Place this logic in the *set* method for the wage variable.

4. The main method must explicitly create a new *PieceWorker* object and assign it to an element of the *employees* array.

5. If you have any questions as you proceed, ask your lab instructor for assistance.

**Follow-Up Question and Activity**

1. Explain the line of code in your PayrollSystemTest’s main method that calls method earnings. Why can that line invoke method earnings on every element of the employees array?

This line can invoke all earnings methods because they are all overridden in their individual classes. Each subclass of employee has an earnings method inherited from the employee superclass. Because each type of employee’s earnings is calculated differently, overriding the earnings methods allows us to call the same method for each employee, but receive different results.

**Please submit this Word document (fill it out) and the .java files you need to create (PieceWorker.java and PayrollSystremTest.java).**

**Analysis:**

(Describe the problem including input and output in your own words.)

For this assignment we were required to use the code found in our text in figures 10.4 – 10.9. This code contains the class definitions for the super class employee, the subclasses for employee types, and the main method for the payroll program. After importing this code we were required to modify it to include a new class PieceWorker, which is a subclass of employee. After implementing the necessary methods, variables, and a constructor for this class we then had to modify the main method to include a new employee object of type PieceWorker and initialize its variables.

The program takes no input from the user and prints out blocks of text that contains information about each employee in the payroll.

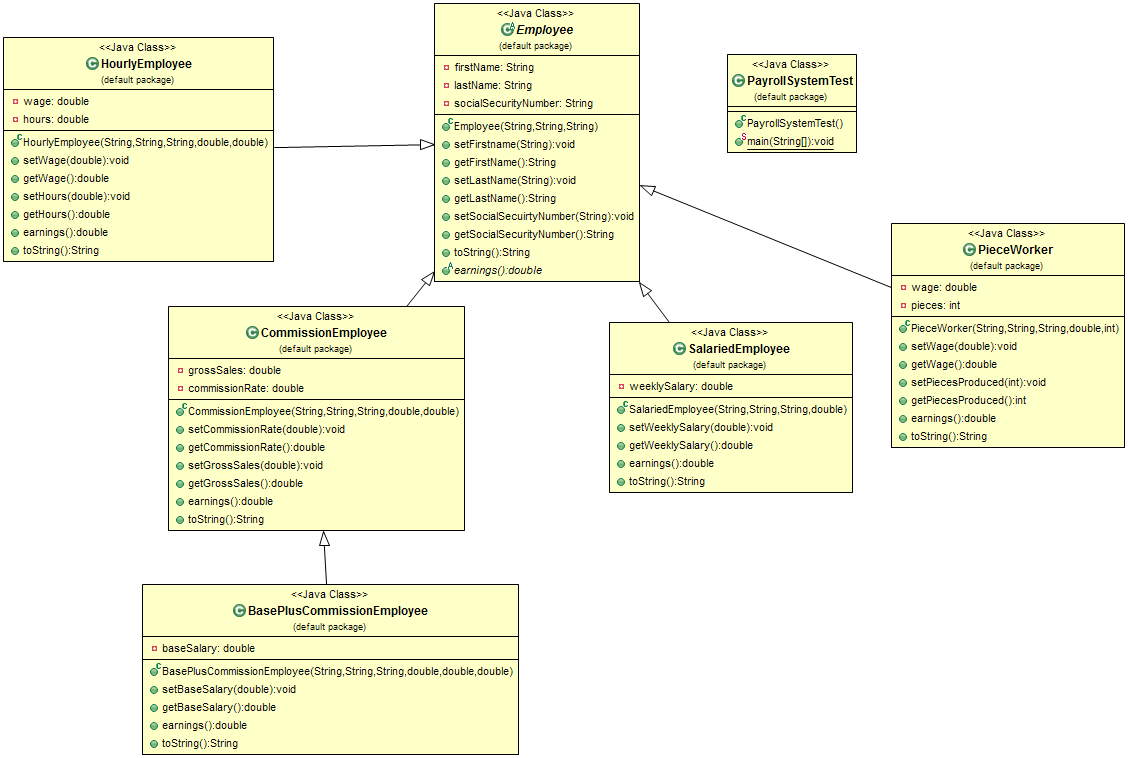
**Design:**

(Describe the major steps for solving the problem. Create a UML diagram to accompany your major steps).

After importing the code from the text, the first step was to create the class PieceWorker. I first declared its variables wage and pieces as a double and int respectively. I was able to determine their data types by the skeleton code for the PieceWorker class in this document and looking at the set and get methods for these attributes. Following this, I created its constructor by having it call the employee superclass and then initializing its own variables. After that I fleshed out the other remaining methods and made sure they had logic checks where they were appropriate.

Once the PieceWorker class was finished I inserted a new employee object of type PieceWorker into the employee array in the main method and initialized its variables to the specified values based on the sample output in this document.

UML diagram follows:



**Testing**: (Describe how you tested, or will test this program)

I tested this program by cross-referencing its output to the sample output in this document and making sure that they matched. Next, to ensure the logic filters in the set methods for the employee subclasses were working, I changed the initializing arguments in the main method to negative values. This caused the program to throw illegal argument exceptions and confirmed the program was checking if the values were valid.

**Submit the following items:**

1. Save this Word file; submit it via the Webcourses Assignments on or before the due date and time.

2. Compile, Run, and Submit your .java file(s) to Webcourses. You must submit the program regardless whether it is complete or incomplete, correct or incorrect.