Assignment 7

CS103

Myers

Concepts tested by this program:

Reading from a file

New concepts tested by this program

Inheritance

Interface

Polymorphism

Abstract classes

Overriding methods

super()

Writing to a file

WEEKLY PAY REPORT FOR THE WIDGET COMPANY

The Widget Company has four types of employees. Managers (who receive a fixed weekly salary), Hourly workers (who receive a fixed hourly wage for up to the first 40 hours they work and "time-and-a-half," i.e., 1.5 times their hourly wage, for overtime hours worked), Commission workers (who receive a $250 plus 5.7% of their gross weekly sales), and Pieceworkers (who receive a fixed amount of money per item for each of the items they produce -- each pieceworker in this company works on only one type of item).

Write a Java program to compute the weekly pay for each employee and the total payroll. You do not know the number of employees in advance. Each type of employee has its own pay code: Managers have paycode 1, hourly workers have code 2, commission workers have code 3 and pieceworkers have code 4. Input will come from a sequential file that has a line for each employee. The first item in the line will always be the paycode. For a paycode of 1, the salary will be the second item. If the pay code is 2, hourly salary is the second item and hours worked is the third item. If the pay code is 3, the second item will be the gross weekly sales. If the pay code is 4, the second item will be the price per piece and the third item will be the number of pieces. The last item on the line will be the worker's ID number.

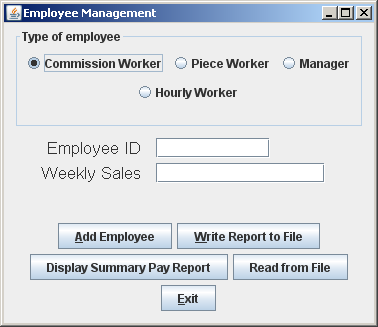
An employee can also be added through the GUI.

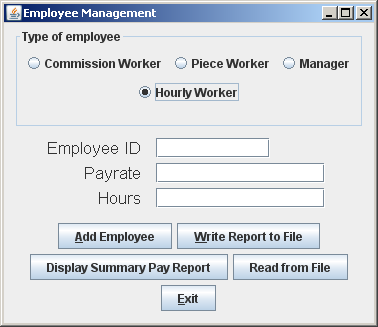
A Summary Pay Report can be sent to the screen in a JOptionPane window or to a sequential file. Each employee number and weekly wage is printed before the summary items. The summary should consist of the total payroll and the number of employees in each category. Use the JFileChooser dialog box to select the input and output file names at run time. Print all dollar amounts with currency formatting. You can assume all information in the file is correct.

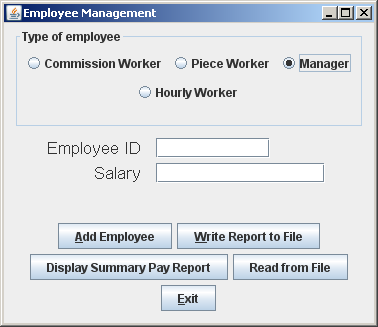
## Specifications

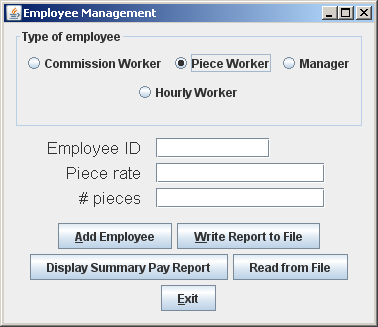
* Create an abstract class called Employee with one abstract method called calculateWeeklyPay which returns a double. This class will have an instance variable for the employee id, a toString method, getters and setters and a constructor.
* Create the following classes for the 4 types of employees: Manager, HourlyWorker, CommissionWorker, PieceWorker. They will contain instance variables appropriate for the type of employee. They will extend from Employee and define the calculateWeeklyPay method. There should also be getter and setters, and any other methods that are needed for your design. The toString method and constructors for these classes will use the super reference to take advantage of the Employee constructor and toString methods. Use finals to represent constants.
* Create an EmployeesManager class that implements the EmployesManagerInterface. It will contain 1 arraylist which holds references to Employee objects (Manager, HourlyWorker, CommissionWorker, PieceWorker all inherit from Employee). It will have a method to addEmployee, calculateWeeklyTotal, retrieve number of hourly workers, commission workers, etc., and any other methods needed for your design. Your toString method will print out the employee number and weekly pay for all employees. It will have a generateWeeklyPayReport method that returns a string with the entire report.
* Create a Driver class which implements the GUI. The driver will read from a file, and print the summary pay report to a file. Use the JFileChooser dialog box to select the input and output file names at run time. The labels and number of fields will change depending on the type of employee that is selected. You can assume all information in the file is correct. Uses an object of EmployeesManager.
* Create Javadoc for your Manager, HourlyWorker, CommissionWorker, PieceWorker and EmployeesManager classes.
* Create a JUnit Test for your Manager class (ManagerTest) and your HourlyWorker class (HourlyWorkerTest).

**Labels and TextFields change depending on type of employee:**

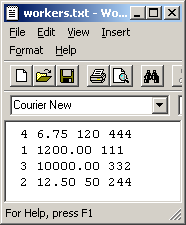




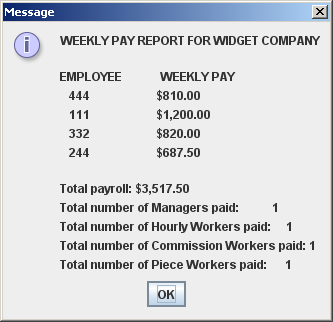




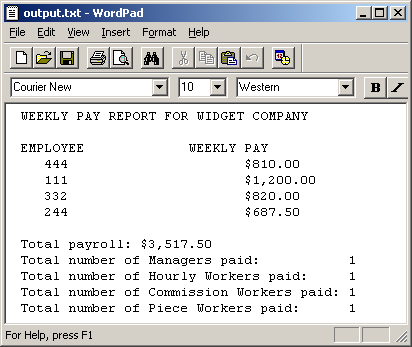
**Input file for test case:**

****

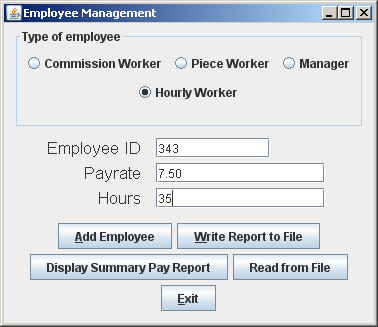
**Display summary pay report:**



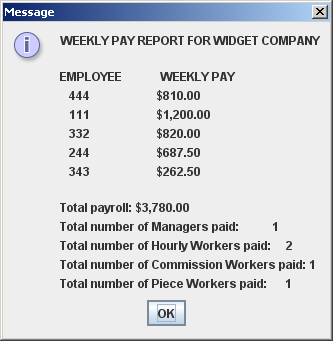
**Write to a file:**



Add an Hourly Worker:



Resulting Pay Report:



**Program Grade Sheet** Assignment #7

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DOCUMENTATION

**Javadoc** for the following classes: 5 pt \_\_\_\_\_

Generate for all classes created by the student

Test Cases 7 pt \_\_\_\_\_

**JUnit Test Class**

Create test classes for Manager and HourlyWorker

**UML Diagram**  3 pt \_\_\_\_\_

PROGRAMMING

Internal class documentation (within source code) using Javadoc 2 pt \_\_\_\_\_

Description of what class does

Author’s Name, @author

Methods commented properly using Javadoc

Description

@param, @return

Compiles and Runs without runtime errors or warnings 2 pt \_\_\_\_\_

Program user interface

Clear to user how data is to be entered 1 pt \_\_\_\_\_

Output is easy to understand 1 pt \_\_\_\_\_

Accuracy – Received correct output

Public tests – Junit test given to you for EmployeesManager 3 pt \_\_\_\_\_

Public tests – your ManagerTest, HourlyWorkerTest 4 pt \_\_\_\_\_

Private tests 5 pt \_\_\_\_\_

Program Details

Data Element Abstract Class – Employee 3 pt \_\_\_\_\_

1. Abstract method calculateWeeklyPay
2. Common instance variable
3. Defines constructor and toString methods

Data Element Concrete Classes – inherit from Employee 4 pt \_\_\_\_\_

1. calculateWeeklyPay defined for all classes
2. additional methods as needed
3. uses super() in definition of constructor and toString methods

Data Manager – Employees 5 pt \_\_\_\_\_

1. implements EmployeesInterface
2. gives definition to all abstract methods
3. Uses an ArrayList to hold Employee objects

GUI Driver 5 pt \_\_\_\_\_

1. Uses a JFileChooser for user to select files
2. Reads from a file
3. Writes to a file
4. Fields and labels change depending on type of employee
5. Weekly pay is formatted to two decimal places

Total 50 pt \_\_\_\_\_

Comments: