Assignment 6

CMSC203

Spring 2018

Concepts tested by this program:

Reading from a file

ArrayList<>

New concepts tested by this program

Inheritance

Interface

Polymorphism

Abstract classes

Overriding methods

super()

WEEKLY PAY REPORT FOR THE Wacky Widgets COMPANY

The Wacky Widgets Company has four types of employees: Managers (who receive a fixed weekly salary), Design 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), Sales workers (who receive a $250 plus 5.7% of their gross weekly sales), and Manufacturing (who receive a fixed amount of money per item for each of the items they produce -- each manufacturer 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. The first item in the report will be the first name, the second item in the line will be the last name, the third item will always be the employee type (Manager, Design, Manufacturing, Sales). For a Manager, the salary will be the fourth item. If the type is Design, hourly salary is the fourth item and hours worked is the fifth item. If the type is Sales, the fourth item will be the gross weekly sales. If the type is Manufacturing, the fourth item will be the price per piece and the fifth 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 is displayed on the screen in a JOptionPane window. 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 FileChooser dialog box to select the input file name at run time. Print all dollar amounts with currency formatting. You can assume all information in the input file is correct.

Allow the user the read in an employee text file (provided), add an employee in the GUI, or remove an employee.

## Specifications

**Data Element - Employee**

* Create an abstract class called Employee with one abstract method called calculateWeeklyPay which returns a double. This class will have an instance variables (one of them is an object of type Position, from Assignment 5), a toString method, getters and setters and at least 2 constructors, one of the constructors must be the default (no-arg) constructor.

**Data Element – subclasses of Employee**

* Create the following classes for the 4 types of employees: Manager, Design, Sales, and Manufacturing. They will contain instance variables appropriate for the type of employee. They will extend from Employee and define the calculateWeeklyPay method. There should be getters 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.

**Enumerated Type – Position**

* Create an enumerated type called Position. The valid values will be MANAGER, SALES, DESIGN, MANUFACTURING. Add any additional methods or instance variables as required by your design.

**Data Structure**

* You will be using an Arraylist within your Data Manager to hold the employee objects.

**Data Manager - Company**

* Create a Company class that implements the CompanyInterface. It will contain an arraylist which holds references to Employee objects (Manager, Design, Sales, Manufacturing all inherit from Employee). It will have methods to addEmployee, removeEmployee, printEmployees, calculateTotalWeeklyPay, retrieve number of Design employees, Sales employees, etc., and any other methods needed for your design. Your toString method will print out the employee number and weekly pay for all employees. There is a generateWeeklyPayReport method that returns a string with the entire report.

**GUI Driver**

* Create a GUI Driver class/classes. The preferred submission will be using JavaFX, although Swing submissions are also acceptable. The GUI will be able to read from a file, add or remove an employee, print a summary of employees, or print a pay report by employee. Use the FileChooser dialog box to select the input file name at run time. The number of fields in the file for one employee will change depending on the type of employee that is selected. The fields in the file are separated by colons (:). You can assume all information in the file is correct. The GUI will use an instance of the Company class.
* The user will have the ability to add additional employees through the keyboard. The labels and textfields will change depending on the type of employee that is selected to be added.
* The user will have the ability to remove an employee. If there are not employees yet added, use a JOptionPane to tell the user to add employees first. If the user does not type the first and last name of the employee to be removed, use a JOptionPane to display the current employees and tell the user to type in a name.
* Display a report of all employees, and a weekly pay report for all the employees
* Note: your GUI need not display an icon.

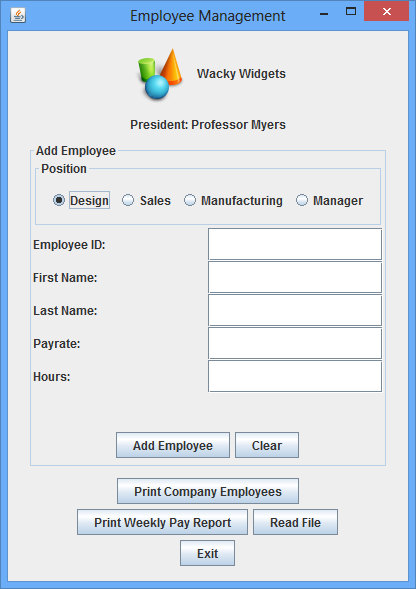
**External Documentation**

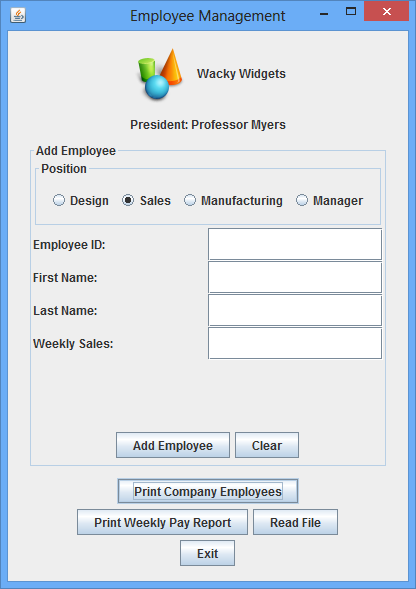
* Provide a UML diagram with all classes and their relationships.
* Create Javadoc for all classes that you have created. Upload your entire doc folder.

**Testing**

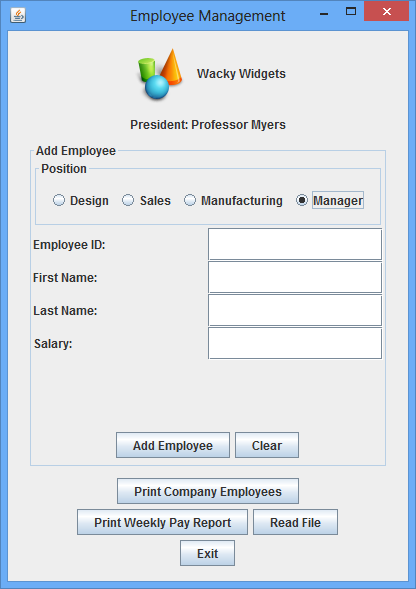
* Ensure that the JUnit tests in CompanyTest.java succeed.
* Implement the STUDENT test methods in CompanyTest.java.
* Include CompanyTest.java with the rest of your submission.

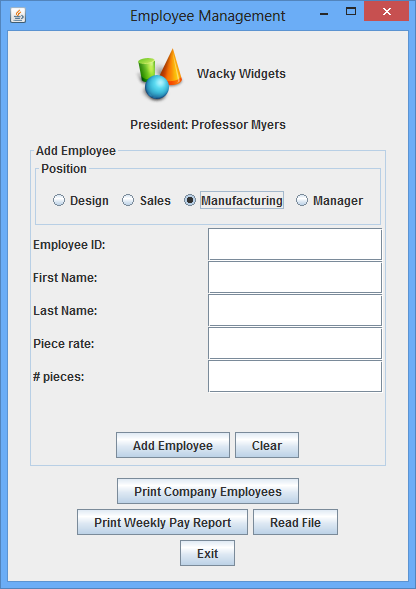
**Labels and TextFields change depending on type of employee:**

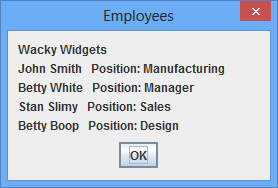
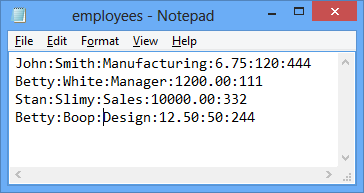




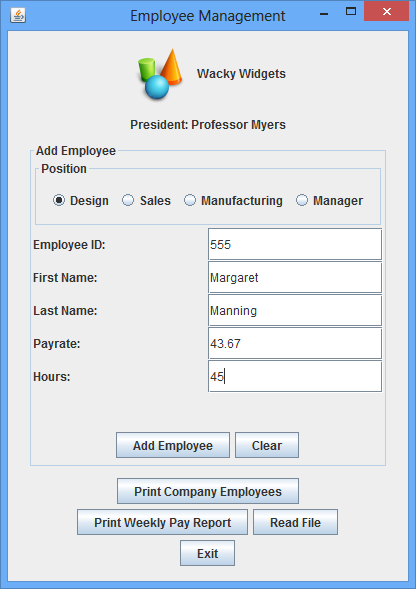
**Input file for test case: Print Company Employees:**



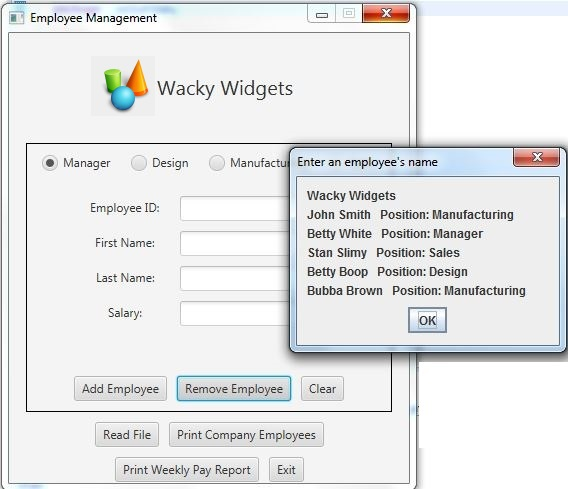




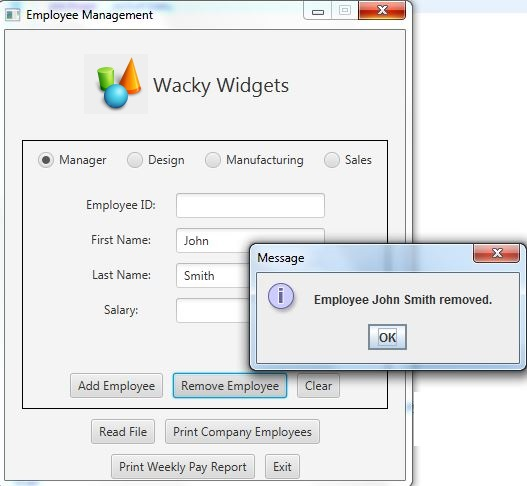
**Display summary pay report: Add a Design Employee:**

**Resulting Pay Report: Remove Employee if no name has been entered:**

**Remove an employee (correctly):** **Result:**



**Deliverables / Submissions:**

**Week 1:** Design – a UML class diagram.

**Week 2:** Submit two compressed files containing the follow (see below):

**Deliverable format:** The deliverables will be packaged as follows. Two compressed files in the following formats:

LastNameFirstName\_Assignment6.zip (a compressed file containing the following)

* + Javadoc (a directory) *a compressed file of the Javadoc doc directory*
  + Source Code (a directory) *contains your .java files from the src directory*
    - File1.java (example)
    - File2.java (example)
    - File\_Test.java (example)

LastNameFirstName\_Assignment6\_**Moss**.zip (a compressed file containing only .java files)

*NO FOLDERS!!*

File1.java (example)

File2.java (example)

**Grading Rubric**

**CMSC 203 Project #6**

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

**Overview:**

There are two parts to the rubric. First, the project is graded on whether it compiles, whether it runs without errors, and whether it satisfies the specifications. These points add up to 100. Second, the score is decremented if various requirements are not met, e.g., no Javadoc, no UML diagram, uses constructs that are not allowed, etc.

**PROGRAMMING** (100 pts)

Compiles 40 pts \_\_\_\_\_

Accuracy

Passes JUnit tests (provided) 10 pts \_\_\_\_\_

Passes JUnit student tests 5 pts \_\_\_\_\_

Passes private instructor tests 10 pts \_\_\_\_\_

Design

Initial design was submitted 7 pts \_\_\_\_\_

Final design and Design Reflection 13 pts \_\_\_\_\_

Execution: runs without errors (either run-time or logic errors) 15 pts \_\_\_\_\_

Possible Sub-total 100 pts \_\_\_\_\_

**REQUIREMENTS** (Subtracts from Programming total)

**Documentation:**

Javadoc is not submitted - 4 pts \_\_\_\_\_

Documentation within source code is missing or incorrect - 4 pts \_\_\_\_\_

Description of what class does is missing

Author’s Name, @author, is missing

Methods not commented properly using Javadoc @param, @return

**Programming Style:**

Incorrect use of indentation, statements, structures - 3 pts \_\_\_\_\_

User interface

Not clear to user how data is to be entered; UI does not follow requirements - 4 pts \_\_\_\_\_

Output is easy to understand - 4 pts \_\_\_\_\_

**Design:**

Final Design does not match implementation - 5 pts \_\_\_\_\_

Data Element Abstract Class – Employee is not used -10 pts \_\_\_\_\_

1. Abstract method calculateWeeklyPay not used
2. Common instance variable/variables not used
3. Constructor and toString methods not used

Data Element Concrete Classes – do not inherit from Employee -8 pts \_\_\_\_\_

1. calculateWeeklyPay not defined for all classes
2. Constructors, getters, setters, toString methods not used
3. Does not use super() in definition of constructor

Data Manager – Company -10 pts \_\_\_\_\_

1. Does not implement CompanyInterface
2. Does not define interface’s abstract methods
3. Does not use ArrayList to hold Employee objects

GUI Driver -8 pts \_\_\_\_\_

1. Does not use a JFileChooser for user to select files
2. Does not correctly read from given file
3. Does not change fields and labels depending on type of employee
4. Does not format weekly pay with currency format

Possible decrements: -60 pts \_\_\_\_\_

Possible total grade: 100 pts \_\_\_\_\_