

CST8132 Assignment 02: Employee Management Tool Prototype

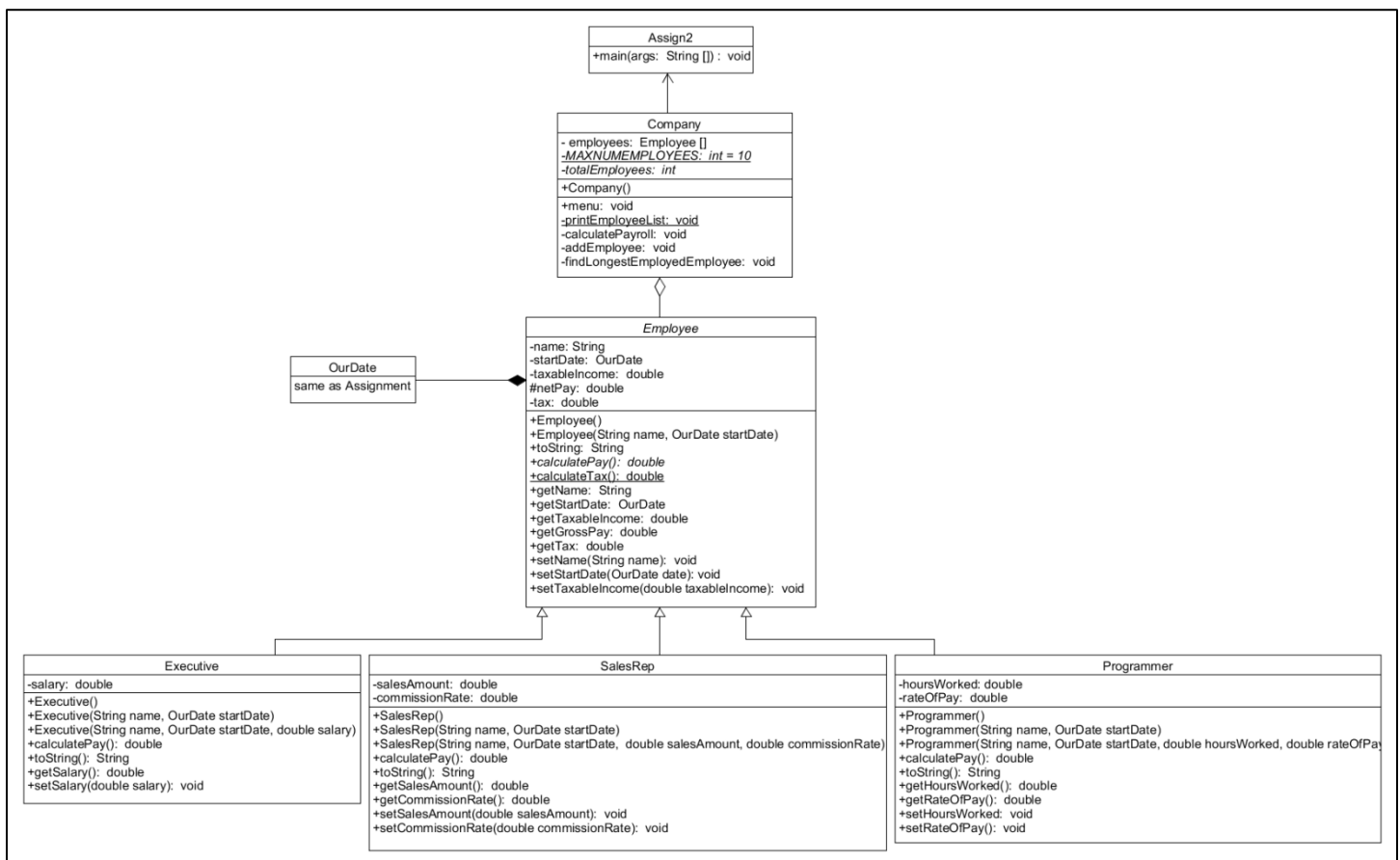
Due: Friday March 11, 2016 by 11:59pm via the Blackboard upload link

Note: The upload link will disappear immediately after the due date/time.

Problem Description

Develop a tool to keep track of a company's Employees. The Employee Management Prototype will be able to add employees, calculate the weekly payroll and the associated taxes, view employees and their employment data, and determine which employee has been employed at the Company for the longest period. Each employee's data will be stored in an Employee array (set maximum size for now to be 10). Every employee will have a name, a start employment date (day, month, year) a weekly pay amount and a weekly tax amount. All date input entered from the keyboard must be fully edited for valid data – you may use your solution for the OurDate class from assignment #1.

Program Design



- **Assign2** contains only a main method, which instantiates one Company reference and calls menu().
- **Company** has a private static int constant which is used to represent the maximum amount of employees, a private static int which is used to keep track of the current amount of employees, an array for Employee references (10 references), one constructor and five methods:
 - menu() - shows the menu, interacts with the user for menu selection, loops till they exit.
 - addEmployee().
 - verifies that the maximum number of Employees is not already hired by the company.
 - Prompts for the employee's name.
 - Prompts for the employee's hiring date.
 - Presents a second level of choices for the user to choose the type of Employee to add (Executive, SalesRep or Programmer).
 - Keeps track of the current total amount of employees.
 - printEmployeeList() - displays information for the current employees.
 - calculatePayroll() - invokes the calculatePay() method for each employee and keeps a running total of the total company pay.
 - findLongestEmployedEmployee() - determines who is currently the longest employed employee based on the results returned from the calculateDays() method in the OurDate class.
- **Employee** is an abstract class. It has an OurDate reference to represent the employee's hiring date, a String for the employee's name, three double fields for the employee's taxable income (the gross pay), the tax to be deducted, and the net pay (after tax pay), three constructors, a "getter" and a "setter" for each field, and three methods:
 - calculatePay() - an abstract method with concrete implementations in each of the subclasses.
 - calculateTax() - in order to add some authenticity to the tax deductions, refer to a reference document of your choice in order to find the Canadian Federal Tax Rates for 2016.
 - toString() - returns a String representation of each class reference.
- **Executive** extends Employee. It has a salary field and three constructors, a "getter" and a "setter" for salary, and overridden methods for both calculatePay() and toString().
- **SalesRep** extends Employee. It has salesAmount and commissionRate fields and three constructors, a "getter" and a "setter" for each field, and overridden methods for both calculatePay() and toString().
- **Programmer** extends Employee. It has hoursWorked and rateOfPay fields and three constructors, a "getter" and a "setter" for each field, and overridden methods for both calculatePay() and toString().
- **OurDate**
 - Reuse OurDate from assignment #1

Notes:

- NOTE 1: Only Company interacts with the user; no other classes should use Scanner or println (etc.).
- NOTE 2: **Do not use** recursion for your method calls - instead use a properly coded repetition structure. You will lose marks if recursion is used
- NOTE 3: **Do not use** any kind of "go-to" statement such as continue, break, System.exit(0), etc... in order to implement program logic. Instead, use a properly coded control structure and test for continuity or true/false conditions with a boolean expression. You will lose marks if any "go-to" statements are used. Appropriate places for "break" statements include using them in a "switch" selection structure.
- NOTE 4: There are a number of ways in which to solve this problem. Originality is encouraged as long as the basic program functionality is implemented - consult with your lab instructor.

Testing Design

The scope of testing required will be limited to only those specified in the UML below. There should only be one assert statement per test case. Each unit test should have an appropriate message if a test fails.

EmployeeCalculateTaxTest

```
+ testCalculateTaxForTaxableIncomeBelowFirstLevelIncome(): void
+ testCalculateTaxForTaxableIncomeBetweenFirstAndSecondLevelIncome(): void
+ testCalculateTaxForTaxableIncomeBetweenSecondAndThirdLevelIncome(): void
+ testCalculateTaxForTaxableIncomeBetweenThirdAndFourthLevelIncome(): void
+ testCalculateTaxForTaxableIncomeAboveFourthLevelIncome(): void
```

Canada's Federal Tax Rates for 2016

Reference: <http://www.cra-arc.gc.ca/tx/ndvdl/s/fq/txrts-eng.html> and <http://www.cra-arc.gc.ca/tx/ndvdl/s/tpcs/ncm-tx/rtrn/cmpltng/ddctns/lrs300-350/323/dctn-eng.html>.

Federal tax rates for 2016

- 15% **on the first** \$45,282 of taxable income, +
- 20.5% **on the next** \$45,281 of taxable income (on the portion of taxable income over \$45,282 up to \$90,563), +
- 26% **on the next** \$49,825 of taxable income (on the portion of taxable income over \$90,563 up to \$140,388), +
- 29% **on the next** \$59,612 of taxable income (on the portion of taxable income over \$140,388 up to \$200,000), +
- 33% of taxable income **over** \$200,000.

Tasks

- ☐ Build the program, do not overlook the package name used for the program classes.
- ☐ Build the JUnit test classes, use a separate package as indicated in the UML.

Submission Requirements

- Place source code files into a zip archive and upload it to Blackboard by the due date.
- Your lab professor will indicate any additional submission requirements to you in the lab

Comments Note

<ul style="list-style-type: none"> At the top of each source code file include the following comment header, adding the needed information: 	<pre>/* File Name: * Course Name: * Lab Section: * Student Name: * Date: */</pre>
<ul style="list-style-type: none"> Classes and class members (class level fields, constructors, methods) should have a brief description as a comment immediately above in the code listing. Example: 	<pre>/* * Represents a Company to enable the * implementation of human resource functionality */ public class Company{</pre>

Grading Rubric

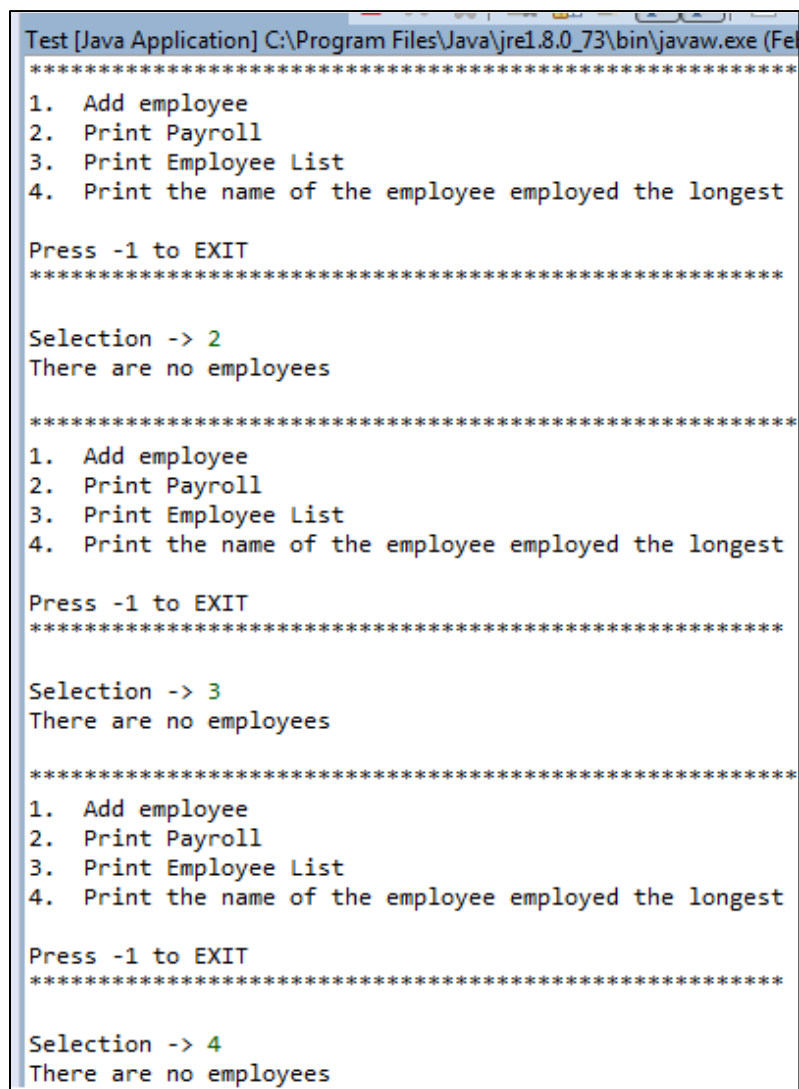
Criteria	Needs Work (0)	Poor (1)	Intermediate (2)	Excellent (3)	Value Scored
Naming	Classes and class members follow no Java naming conventions.	Java naming conventions are not well followed.	Classes and class members follow Java naming conventions with tiny inconsistencies.	All classes, class members are named following Java naming conventions, consistently and perfectly.	
Comments (Javadoc not required Assign 2)	Comments missing or incorrect.	Many classes and / or class members missing meaningful comments	Very few classes and / or members missing comments, comments are meaningful.	Nearly everything is commented, comments are meaningful, brief, well written.	
Citations	No referenced work cited when it needed to be. See Algonquin College Policy AA20 on Plagiarism.	References and citations present but not APA style, e.g. only the URL provided.	References and citations loosely follow APA style	References and citations closely follow APA style	
Compiles	Program does not compile, too many syntax mistakes for the professor to track or debug without major re-write.	Program does not compile, has several syntax mistakes	Program does not compile, has a few small syntax mistakes	Program compiles	
Execution	Program is missing much functionality. For example program starts to run but does not work correctly.	Program is missing much code, and much of the required concepts.	Program demonstrates most of the concepts, some parts left out.	Program demonstrates understanding and application of concepts notably manipulation of a 1 dimensional array, and composition.	
Concepts: Inheritance, polymorphism, abstract classes/methods.	Concepts are not demonstrated, program does not follow UML design.	One or two concepts demonstrated, program loosely follows UML design	Program demonstrates most of the concepts, closely follows UML design	Program demonstrates understanding and application of concepts notably inheritance, polymorphism, abstract and follows UML design exactly.	
Unit Tests	Unit tests omitted or largely incomplete	Unit tests work but code is not well organized (within each @Test method)	Unit tests work and each @Test method has well organized code	Unit tests work, each @Test method has well organized code, and meaningful variable names used to improve readability.	
Sub-total Max(21):					
Deductions for recursion or inappropriate use of "go-to" statements Max(-10):					
Total:					

Notes on citations and references

- Please do not cite or reference other students, ask for the original sources from them and cite and reference those instead
- You will not get credit for an assignment or project if large portions are copied directly from other sources, even if you cite and reference the source(s) correctly. Assignments are to be your own original work; other works can be used for help in solving problems or as small pieces of your larger program. Determinations on this are up to the discretion of the professor, if in doubt check with your professor.
- Note: One exception to this is in the case of lecture and lab handouts, and code samples posted to Blackboard. You are free to use these as a starting point just cite + reference them as a personal communication from your professor e.g. ***your professor name here*** (2016) personal communication.

Sample Program Run Screen Captures (green is user input):

Figure 1: Currently No Employees



```
Test [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (Fe...
*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection -> 2
There are no employees

*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection -> 3
There are no employees

*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection -> 4
There are no employees
```

Figure 2: New Executive

```
Press -1 to EXIT
*****

Selection -> 1
Enter new Employee Information:
Name: Jimmy Jones
Hiring Date:
Year: 2016
Month: 1
Day: 30
What kind of employee:
1. Executive
2. Sales Rep
3. Programmer
1
Salary:
145000
*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection ->
```

Figure 3: New Sales Rep.

```
*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection -> 1
Enter new Employee Information:
Name: Sarah Smith
Hiring Date:
Year: 2015
Month: 12
Day: 2
What kind of employee:
1. Executive
2. Sales Rep
3. Programmer
2
Commision Rate:
0.1
Total Sales:
35000
*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection ->
```

Figure 4: New Programmer

```
*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection -> 1
Enter new Employee Information:
Name: John Doe
Hiring Date:
Year: 2015
Month: 12
Day: 1
What kind of employee:
1. Executive
2. Sales Rep
3. Programmer
3
Hourly Wage:
25
Total Hours Worked:
40
*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection ->
```

Figure 5: List Employees and Longest Employed Employee

```
<terminated> Test [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (Feb 20, 2016, 1:32:41 PM)
*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection -> 3
Name: Jimmy Jones      Date of Hire: 2016/1/1
Name: John Doe         Date of Hire: 2015/12/1
Name: Sarah Smith      Date of Hire: 2015/12/2
*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection -> 4
Employee employed for the longest: John Doe for 0 years and 81 days
*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****
```

Figure 6: Print Payroll

```
<terminated> Test [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (Feb 20, 2016, 1:32:41 PM)
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection -> 2
Name: Jimmy Jones      Week Total: 2788.46      Salary: 145000.00
Name: John Doe         Week Total: 1000.00      Hours Worked: 40.00   Hourly Wage: 25.00
Name: Sarah Smith      Week Total: 3500.00      Sales Amount: 35000.00 Commission Rate: 0.10

TOTAL PAYROLL: $7288.46
*****
1. Add employee
2. Print Payroll
3. Print Employee List
4. Print the name of the employee employed the longest

Press -1 to EXIT
*****

Selection -> -1
Goodbye
```