COSC 4301 – MODERN PROGRAMMING
Project 2 – Object–Oriented Programming

A company pays its employees on a weekly basis. The company wants you to write an application that performs its payroll calculations polymorphically. An abstract class, **Employee**, attached is used to represent the general concept of an employee.

The employees are of five types described below. Write a public class for each employee type that inherits the Employee class.

PieceEmployee — For employees whose pay is based on the number of pieces of merchandise produced. Class PieceEmployee 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 PieceEmployee that calculates the employee’s earnings by multiplying the number of pieces produced by the wage per piece.

SalariedEmployee — For Employees who are paid a fixed weekly salary, this class should contain a private weeklySalary instance variable, and should implement method earnings to return the weeklySalary.

HourlyEmployee — For Employees who are paid by the hour and receive overtime pay for all hours worked in excess of 40 hours per week, this class should contain wage and hours instance variables (private), and should implement method earnings based on the number of hours worked.

CommissionEmployee — For Employees who are paid by commission, this class should contain grossSales and commissionRate instance variables (private), and should implement method earnings to return grossSales * commissionRate.

BasePlusCommissionEmployee — For Employees who are paid a base salary and commission. This class should contain a private instance variable named baseSalary. This class should inherit the CommissionEmployee class and use the earnings method in CommissionEmployee class and the baseSalary to calculate the weekly salary. The formula for calculating the earnings is baseSalary + grossSales * commissionRate.

The UML class diagram attached shows the inheritance hierarchy for the polymorphic employee-payroll application. Abstract class name **Employee** is italicized — a convention of the UML.

Create a class named **Project2,** attached that creates an array of Employee variables to store references to objects of each concrete class in the Employee hierarchy. For each Employee, display its String representation and earnings.

Study the Employee abstract class and the Project2 class and understand them before starting the project. Do not modify them. You will not receive credit for this

project if you do. You are allowed to modify line 91 of the Project2.java file. Line 91 is shown below.

```
   System.out.println("Name:    <Put your full name here>");
```

All classes in this project must be public, non-static and not nested in other classes.

Every method in your program should be limited to performing a single, well-defined task, and the name of the method should express that task effectively.

Run your program and copy and paste the output to a file named **Project2-Output.txt**. Create a folder named, **<YourFullName>_Project2**. Copy your source codes and the output file to the folder. **Zip the folder, as a ".zip" file, and upload it to Blackboard**.

Before you upload your project to Blackboard:

- Ensure that your code conforms to the style expectations set out in class and briefly discussed below.

- Make sure your variable names and methods are descriptive and follow standard capitalization conventions.

- Put comments wherever necessary. Comments at the top of each module should include your name, file name, and a description of the module. Comments at the beginning of methods describe what the method does, what the parameters are, and what the return value is. Use comments elsewhere to help your reader follow the flow of your code. See the Orientation Project for more details.

- *Program readability and elegance are as important as correctness.* After you have written your method, read and re-read it to eliminate any redundant lines of code, and to make sure variables and methods names are intuitive and relevant.

Read the assignment very carefully to ensure that you have followed all instructions and satisfied all requirements. **You will not get full credit for this project if it is not written as instructed even if it works as expected.**

Sample partial output:

```
Employees processed polymorphically:

...

Hourly Employee: Karen Price
Social Security Number: 222-22-2222
```

```
Hourly wage: $16.75; Hours Worked: 40.00
Weekly Salary: $670.00

...
```