CSC 7200
Week 2 Programming Assignment

Inheritance and Polymorphism

The objective of this programming assignment is to explore the use of inheritance in Java and to see how polymorphism works with inheritance in Java. This assignment involves writing three classes, plus a test class.

The base class is an `Employee` class which should contain a couple of attributes common to all employees and a fundamental method to calculate pay. The two derived classes are a `CommisionEmployee` class, which adds payment of a sales commission as part of the pay calculation, and a `UnionEmployee` class, which adds overtime payment and union dues as part of the pay calculation. The `TestEmployees` class will be a program (i.e., a class that contains a `main()` method) which will include a method that accepts an `Employee` base class reference and demonstrates polymorphic behavior.

The details of the three classes to be implemented are as follows:

1. `Employee`: An Employee class contains a name, a department where the employee works, and an hourly rate of pay. An explicit value constructor should be provided to set all three values when an Employee object is created. There should be mutator methods to set the values of the department and the pay rate. There should be one accessor method which returns a string containing the name and the department in which the employee works. Finally, there should be a weekly pay method that takes an integer parameter for the number of hours worked and returns the weekly pay. If the number of hours worked is less than 40 hours, the pay is the number of hours times the rate. If the number of hours worked is 40 or more, the pay is 40 times the rate.

2. `UnionEmployee`: The Union Employee class inherits from the Employee class. A Union Employee contains a dues variable which holds the amount of dues a Union Employee has withheld from their paycheck each week. An explicit value constructor should be provided to set all three values from the base class along with the dues value. There should be a mutator method provided to set the dues variable. The base class weekly pay method should be overridden because a Union Employee gets 1.5 times the rate for any hours over 40 per week. This method should use the base class method to calculate pay for first 40 hours and then add the overtime amount. Also the weekly pay is reduced by the amount of dues withheld.

3. `CommisionEmployee`: The Commission Employee class inherits from the Employee class. A Commission Employee contains a commission rate and a sales amount variable which are used as part of the pay calculation. An explicit value constructor should be provided to set all 3 values of the base class along with the commission rate variable. There should be mutator methods for the commission rate and the sales amount. The base class weekly pay method should be overridden because the Commission Employee gets the base employee pay plus the commission rate times the sales amount. This method should use the base class weekly pay method to calculate the hourly part of the pay.

4. `TestEmployees`: The test employees program needs to create a Union Employee object and a Commission Employee object. The test program must contain a display method which takes a base class Employee object reference along with the number of hours worked by the employee. The display method should use the base class method to get the employee name and department info and output that information. The display method should also use the base class method to get the weekly pay info for the employee object and display that information. The test program should pass the Union Employee object and the Commission Employee object to the display method along with the number of hours each employee has worked. It should test the payroll calculation for the number of hours worked to be less than 40, 40, and greater than 40 hours for each employee type. The output seen should demonstrate polymorphic behavior, that is the base class Employee reference to a Union Employee object elicits Union Employee pay calculations, and the base class Employee reference to a Commission Employee elicits Commission Employee payroll calculations.

Deliverables

- Screenshots of the running program
- Source Code for all classes
- Paste Screenshots + Source Code into a single DOC/DOCX file

Lab Grading Criteria

```
Program Source Code          120 pts.
Screenshots                  30 pts.
-----------------------------------
Total   150 pts.
```

*** Program1:
- Don't have a base employee class: -15
- Don't have a derived commissioned employee class: -9
- Don't have a derived union employee class: -9
- Don't have a test class: -15

- Base employee class doesn't have an explicit value constructor: -6
- Base employee class doesn't have mutator methods for department and payrate: -6/each
- Base employee class doesn't have an accessor method for name and department: -6
- Base employee class doesn't have a weekly pay method: -6
- Base employee class's weekly pay method doesn't take number of hours parameters: -6
- Base employee class's weekly pay method doesn't return number of hours times the rate: -6

- Union employee doesn't inherit from base employee class: -9
- Union employee doesn't have an explicit value constructor: -6
- Union employee doesn't override weekly pay method: -9

- Commission employee doesn't inherit from base employee class: -9
- Commission employee doesn't have an explicit value constructor: -6
- Commission employee doesn't override weekly pay method: -9

- Test class doesn't accept a base class reference: -6
- Test class doesn't create a Union Employee object: -6
- Test class doesn't create a Commission Employee object: -6
- Test class doesn't have a display method: -9
- Test class's display method doesn't take base employee class object reference: -6