**CS1180 Project 5  
Inheritance and Polymorphism**

The objective of this programming assignment is to experience the use of inheritance in Java and to see how polymorphism works with inheritance in Java.

The assignment involves writing three classes, plus a test class. The base class is a TaxableWorker class which contains a couple of attributes and methods common to all workers. The first derived class is a StateTaxableWorker which adds state tax information to a TaxableWorker. The second derived class is a LocalTaxableWorker which adds local tax information to a StateTaxableWorker. The test program will be structured to include a method which accepts a base class reference parameter and demonstrates polymorphic behavior.

**NOTE: None of the first three classes below do any user input or console output! User input and console output are only done in the test program code!**

**NOTE: ALL the member variables of these classes MUST be PRIVATE! NO static variables are allowed in any of these classes.**

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

1. A TaxableWorker contains a **name**, an hourly **pay rate** (ie. 12.50/hr), and a federal **tax rate** (ie. 0.25). An **explicit value constructor** must be provided to set all three values. There must be mutator (setter) methods to change the values of the pay rate and the tax rate. There must be a **workerInfo** method that returns a **string** containing the name, hourly pay rate, and federal tax rate. There must be a **grossPay** method that takes the number of **hours worked** as a **parameter** and calculates the gross pay (hours \* payRate) and returns that as a double. There must be a **taxWithheld** method that takes a **gross pay** amount as a **parameter** and calculates the tax withheld (grossPay \* taxRate) and returns that as a double.

**Private Member Variables:**

name, payRate, taxRate

**Public Methods:**

a constructor – takes 3 parameters

two mutators – each takes 1 parameter – no returned values

workerInfo – takes no parameters – returns a string

grossPay – takes one parameter – returns a double

taxWithheld – takes one parameter – returns a double

Objects of this class will also be able to save their data to a file using a PrintWriter, and construct themselves by reading data from a file using a Scanner. The following two methods will accomplish this.

**Public Methods for File Saving and Initialization:**

a constructor – takes a Scanner object (for file reading) parameter – initializes member variables by reading from the scanner object

saveData – takes a PrintWriter object – saves the three member variables to the file one per line

**IMPORTANT**: The simplest way to save member variable data will be to use the println method of the PrintWriter. Call println with one member variable at a time so that only one item appears in the file per line. When using the Scanner object to read the data items from the file, you can use the nextLine method to read all the **strings** that were stored. **After reading a number from a file, make sure to call the nextLine method to throw away the end of line following the number that was read. Otherwise a subsequent attempt to read a string using nextLine will return the empty line and screw things up!!!**

**Your class can NOT contain any other member variables or methods!**

1. The StateTaxableWorker class inherits from the TaxableWorker class. This class adds attributes for the **state name** and the **state tax rate (ie. 0.05)**. An **explicit value constructor** must be provided to set the three values of the base class plus the values for the state name and state tax rate. A mutator method must be provided to change the value of the state tax rate. The base class **workerInfo** method must be **overridden** to include all the base class worker information plus the state name and state tax rate information. The base class **taxWithheld** method must be overridden to include the base class tax withheld plus the state tax withheld (gross pay \* state tax rate).

**Private Member Variables:**

stateName, stateTaxRate

**Public Methods:**

a constructor – takes 5 parameters

one mutator –takes 1 parameter – no returned values

workerInfo – takes no parameters – returns a string

taxWithheld – takes one parameter – returns a double

You must also provide a constructor which takes a scanner object, and override the saveData method of the base class which takes a PrintWriter object.

**Public Methods for File Saving and Initialization:**

a constructor – takes a Scanner object parameter – must call the base class constructor first passing the scanner, then initializes member variables by reading from the scanner object

saveData – takes a PrintWriter object – must call the base class method first passing the PrintWriter, then saves the member variables to the file one per line

**Your class can NOT contain any other member variables or methods!**

1. The LocalTaxableWorker class inherits from the StateTaxableWorker class. This class adds attributes for the **city name** and the **city tax rate** (ie. 0.025). An **explicit value constructor** must be provided to set the three values of the base class plus the two values for the state name and state tax rate plus the city name and the city tax rate. A mutator method must be provided to change the value of the city tax rate. The base class **workerInfo** method must be overridden to include all the state worker information (which includes base class info) plus the city name and city tax rate information. The base class **taxWithheld** method must be overridden to include the state tax withheld (which includes the base tax withheld) plus the city tax withheld (gross pay \* city tax rate).

**Private Member Variables:**

cityName, cityTaxRate

**Public Methods:**

1 constructor – takes 7 parameters

one mutator –takes 1 parameter – no returned values

workerInfo – takes no parameters – returns a string

taxWithheld – takes one parameter – returns a double

You must also provide a constructor which takes a scanner object, and override the saveData method which takes a PrintWriter object.

**Public Methods for File Saving and Initialization:**

a constructor – takes a Scanner object parameter – must call the base class constructor first passing the scanner, then initializes member variables by reading from the scanner object

saveData – takes a PrintWriter object – must call the base class method first passing the PrintWriter, then saves the member variables to the file one per line

**Your class can NOT contain any other member variables or methods!**

1. The test program needs to create a TaxableWorker object, a StateTaxableWorker object, and a LocalTaxableWorker object. These three objects must be stored in an ArrayList<TaxableWorker> object. The test program must contain a **display method** which takes a base class TaxableWorker object reference along with the **number of hours worked** by the worker. The display method uses the **workerInfo** method to get the name and tax information about the worker and output that information. The display method should also use the **grossPay** method to get the pay info for the worker and display that information. Then the display method should use the **taxWithheld** method to output the total amount of tax withheld for the worker. All monetary values should be output with 2 digits after the decimal point.

The test program should pass the three objects from the ArrayList<TaxableWorker> in to the display method (one at a time) along with the number of hours each worker has worked. The output seen demonstrates polymorphic behavior, ie. the base class TaxableWorker reference to a StateTaxableWorker object elicits StateTaxableWorker information and correct tax withheld, and the base class TaxableWorker reference to a LocalTaxableWorker object elicits LocalTaxableWorker information and correct tax withheld.

1. The last part of the test program will be to save the object data to a file and then read the object data back from the file. Add a static method to the test program to save the worker data to a file. This method should take an ArrayList<TaxableWorker> object as a parameter. In that method, create a PrintWriter object for writing data to a file. For each object in the array list, determine which class that object belongs to. This can be done using the instanceof operator as follows:

if(workerObject instanceof LocalTaxableWorker)

For that object, output a line to the file identifying which class the object belongs to as follows:

TW – TaxableWorker STW – StateTaxableWorker LTW – LocalTaxableWorker

Then call the saveData method on the object, passing in the PrintWriter object so the object can write its data to the file.

Add a method to load the worker data into an ArrayList<TaxableWorker> object from a file. Create the array list object which will be returned as the result of this method. Create a scanner object to read the file. While there is data to process from the file, read a line to get the type of object stored. Create an object of that object type, passing in the scanner object so the object can read the data it needs from the file. Store the object into the array list.

The main method should call the function to save the worker data from the existing array list to a file. Then call the function to read the data from a file and get a new array list of objects. Then process this new array list with the display method to show that these three worker objects contain the same data as the original.

Example Output:

Employee: Bob Jones, PayRate = $15.00, TaxRate = 15.00%

Gross pay for 40 hours is $600.00, tax withheld is $90.00, net pay is $510.00

Employee: Jim Gordon, PayRate = $18.00, TaxRate = 17.50% | State: New York, StateTaxRate 5.00%

Gross pay for 40 hours is $720.00, tax withheld is $162.00, net pay is $558.00

Employee: Bruce Wayne, PayRate = $25.00, TaxRate = 27.50% | State: New York, StateTaxRate 6.00% | City: Gotham, CityTaxRate 3.50%

Gross pay for 40 hours is $1000.00, tax withheld is $370.00, net pay is $630.00

Process the employee list read from the file.

Employee: Bob Jones, PayRate = $15.00, TaxRate = 15.00%

Gross pay for 40 hours is $600.00, tax withheld is $90.00, net pay is $510.00

Employee: Jim Gordon, PayRate = $18.00, TaxRate = 17.50% | State: New York, StateTaxRate 5.00%

Gross pay for 40 hours is $720.00, tax withheld is $162.00, net pay is $558.00

Employee: Bruce Wayne, PayRate = $25.00, TaxRate = 27.50% | State: New York, StateTaxRate 6.00% | City: Gotham, CityTaxRate 3.50%

Gross pay for 40 hours is $1000.00, tax withheld is $370.00, net pay is $630.00

Example output file:

TW

Bob Jones

15.0

0.15

STW

Jim Gordon

18.0

0.175

New York

0.05

LTW

Bruce Wayne

25.0

0.275

New York

0.06

Gotham

0.035

Development Strategy

There are a lot of pieces to this project. This is how I would recommend attacking this project.

1. Write the TaxableWorker class. Don’t worry about writing the code for the 2 functions to save and load data to / from a file. Leave the bodies of those two methods blank.
2. Test program – write the display method. Create and test a TaxableWorker object.
3. Write the StateTaxableWorker class using inheritance. Don’t implement the save and load methods yet.
4. Test program – Create and test a StateTaxableWorker object.
5. Write the LocalTaxableWorker class using inheritance. Don’t implement the save and load methods yet.
6. Test program – Create and test a LocalTaxableWorker object.
7. Test program – incorporate the ArrayList<TaxableWorker> into the test program.
8. Implement the saveData methods of all three classes starting with the TaxableWorker class.
9. Add the method to the test program to save the data for the array list of worker objects. Verify that the file is created with all the data from the three worker objects.
10. Implement the constructor that takes a scanner object for the three classes starting with the TaxableWorker class.
11. Add the method to the test program that creates a new ArrayList<TaxableWorker> from a file of worker data. Verify that the array list created contains the same worker data as the original list by using the display method as before.

**Rubric:**

* **All member variables are private – no static variables: 20% deduction if this is violated!**
* **TaxableWorker class (20 pts)**
  + **3 member variable, 8 methods**
* **StateTaxableWorker class (20 pts)**
  + **2 member variables, 6 methods**
  + **5 methods make use of super to access base class constructors or methods**
* **LocalTaxableWorker class (20 pts)**
  + **2 member variables, 6 methods**
  + **5 methods make use of super to access base class constructors or methods**
* **Test program** 
  + **Main plus 3 methods (5 pts)**
  + **Display of all types of worker data is correct (10 pts)**
    - **Must be well organized – does not need to exactly match example output format**
  + **File contains all data from all three types of objects (5 pts)**
    - **Not required to exactly match example file organization**
  + **Display of all data from objects created from file data must match data output from original objects (10 pts)**
* **Comments and style (10 pts)**
  + **All classes and methods have proper comments!**