**Lab 8 Part 1** (12 points)

There are several learning objectives to this assignment

* Creating Classes that use Inheritance
* Creating Interfaces and Abstract Classes
* Creating Classes that implement Interfaces and Abstract Classes
* Creating advanced algorithms
* Using NumberFormat

import java.text.NumberFormat;

import java.util.Locale; //default is US and do not need to import java.util.Locale. If you want to change the country, import java.util.Locale and put the appropriate country as an argument in .getCurrencyInstance(<country>).

ExampleNumberFormat foo = NumberFormat.getCurrencyInstance();

String out = foo.format(12345.678); // updates out to the String $12,345.68 (NOTE: will round up)

This part of the assignment requires the coder to extend the Person class provided (Person.txt is provided in Blackboard) to create an Employee class. Employee class have a public constant called CURRENT\_YEAR = 2012 and has two additional private instance variables: (1) an integer called hireDate and (2) a String called idNum. You will need to create two public methods call equals(Object o) and getServiceYears() that calculates seniority.

equals() is based on idNum that is a unique key in the PeopleSoft database. getServiceYears() uses CURRENT\_YEAR and hireDate and calculates the difference. You will need to define getters and setters for hireDate and idNum.

toString() prints “Name: “ with name on line1, “ID Num:” with idNum on line2, “Year Hired: “ that prints out year employee was hired and “Years of Service:” that prints our years of service on line 3.

Additionally, there are two classes that inherent (extend) Employee class called FullTime and Adjunct. FullTime has an instance variable called salary that is a double. Adjunct has two instance variables called hrs and hrRate. You will need to create two public methods for FullTime and Adjunct called toString() and getSalary().

toString() for FullTime prints “Name: “ with name on line1, “ID Num:” with idNum on line2, “Year Hired: “ that prints out year employee was hired and “Years of Service:” that prints our years of service on line 3, and “Annual Salary” that prints out Annual Salary and “Monthly Salary:” that prints ***monthly salary*** on line 4. Use NumberFormat to limit to 2 decimal places

toString() for Adjunct prints “Name: “ with name on line1, “ID Num:” with idNum on line2, “Year Hired: “ that prints out year employee was hired and “Years of Service:” that prints our years of service on line 3, and “Hours” that prints out hours, “Hourly Rate” that prints out hourly rate, and “Monthly Salary” that is based on hours multiplied by hourly rate and divided by 12 months as ***monthly salary*** on line 4. Use NumberFormat to limit to 2 decimal places.

*NOTE: when using .toString(), you do not need to use dot-notation. For example, if you had an Employee object called fred, the statement System.out.println(fred); prints out the fred object (in this case an Employee object), per the Employee .toString().*

getSalary() returns salary for FullTime and computes salary for Adjunct by multiplying hrs x hour rate.

You will need to do the following:

1. Create Employee, FullTime, Adjunct classes including appropriate constructors, getters and setters as well as a driver that tests your classes.

Driver Details

1. Create a FullTime object called fred passing in the following arguments to an appropriate FullTime constructor - "Flinstone, Fred", 2005, "BR-1", 65000.12.
2. Create an Adjunct object called barney passing in the following arguments to an appropriate Adjunct constructor – “Rubble, Barney", 2006, "BR-2", 320, 48.55);
3. Create a default FullTime object called wilma.
4. Use an appropriate setter to update the wilma objects name to “Flintstone, Wilma"
5. Use an appropriate setter to update the wilma object idNum to “BR-3"
6. Use an appropriate setter to update the wilma object hireDate to 2009
7. Use an appropriate setter to update the wilma object salary to 48123.25
8. Create a new Employee object called betty passing in the following arguments to an appropriate Employee constructor - "Rubble, Betty", 2011, "BR-4"
9. Create a default FullTime object called wilma2 passing in the following arguments to an appropriate FullTime constructor - "Slate, Wilma", 2009, "BR-3", 48123.25
10. Create a new Person [] called staff initialized to the following object - fred, barney, wilma, betty, wilma2. NOTE – Does this work? If so, how. There are different types of objects.
11. Use a for each loop to print out each of the staff (created in above step) objects
12. Based on the equals() description above, determine if wilma and wilma are the same person
13. After confirming that wilma and wilma2 are the same objects, use an appropriate setter to update the wilma object name to “Slate, Wilma”
14. Use a for loop to print out staff [0] thru staff[3]. //Simulates deleting staff[4] wilmas object

**Submitting your work**

For all labs you will need to provide a copy of all .java files. **DO NOT PROVIDE .class files. I cannot grade, what I cannot read.** In addition to your .java files, you will need to produce a pix of the screen output in .png or .jpg format for each project that covers all use cases in the lab. For persons using Windows 7 and above OS, you can use the built in snipping tool. Mac OS users, you can see how to take screenshots using the following url - <http://www.wikihow.com/Take-a-Screenshot-in-Mac-OS-X>

You will need to zip your files into a single container. **DO NOT USE .rar for Mac OS.** Submit appropriate .java files and screenshots to show input and output in either .png or .jpg format.