**Lab Assignment – 10   
Total Points: 100  
Due Date: Dec 3rd, 2014 (Wednesday), 11.59 pm**

**Inheritance and Polymorphism**

A company pays its employees on a weekly basis. The employees are of four types: Salaried employees are paid a fixed weekly salary regardless of the number of hours worked, hourly employees are paid by the hour and receive overtime pay (1.5 \* wage) for all hours worked in the excess of 40 hours, commission employees are paid a percentage of their sales and base-commission employees receive a base salary plus a commission based on their gross sales. The company wants to implement a java application that performs its payroll calculations polymorphically. (50 points)

**Employee Class**

**CommissionEmployee**

**HourlyEmployee**

**SalariedEmployee**

**BasePlusCommissionEmployee**

***Employee hierarchy UML Class diagram***

1. Write a Java class – Employee that encapsulates an employee’s first name, last name, SSN. Implement the appropriate get and set methods along with toString( ) method and equals( ) method. Also provide an abstract method called getEarnings( ) with return type of double type.   
   public double getEarnings ( )

**~~{~~**

return 0.0 ;

**~~}~~**

*Note: I will provide instructions to modify the class and method during the class on Dec 1..*

1. Create a Java class – SalariedEmployee which is a special type of Employee with the following additional attribute: weeklySalary. Provide appropriate get, set and equals( ), toString( ) methods. Implement a newer version of getEarnings( ) method that returns the earnings of SalariedEmployee: weeklySalary.
2. Create another Java class – HourlyEmployee which is another special type of Employee with the additional attributes: wage and hours. Provide appropriate set and get methods along with equals( ), toString method. Also implement a newer version of getEarnings( ) method that reflects earnings of HourlyEmployee. (Note: take into the account the rules for calculating the earnings for hourly employee as stated earlier in the problem description)
3. Create Java class – CommissionEmployee, another type of employee with attributes: grossSales and commissionRate. Provide set, get and equals( ), toString( ) methods. Provide a newer version of getEarnings( ) method that returns the earnings as (commissionRate \* grossSales).
4. Implement a specialized type of CommissionEmployee called BasePlusCommissionEmployee class which has an additional attribute: base salary. Provide get, set and equals( ), toString( ) methods. Implement the getEarnings( ) method that returns the earnings as (commissionRate \* grossSales) + baseSalary.

***Note for all above User Defined Classes:***

* Provide appropriate validation code so the right values get populated in the instance variables. For example, the payrate should not be negative.
* Make sure to provide JavaDoc style comments.

Write a Java application (Client) program with a static method called generateEmployees( ) that returns a random list of 10 different types of Employee objects. You could either use an array or an ArrayList to store the employee objects that will be returned. Use a *for loop* to populate randomly different types of employee objects with some random data. You could possibly think a range of values like 1 – 4. If random value is 1, create a HourlyEmployee object with some randomly generated data, if 2, SalariedEmployee object with some random data and so on. I would leave it to your ingenuity to generate and populate these different Employee objects. As these objects are generated add them to your data structure (array or ArrayList that you are using). Finally the method returns this data structure.

In the same application class, implement the main( ) method. Call the generateEmployees( ) static method and using a for loop to print the details of each of the employee along with their earnings on the output window.

(70 points)

1. For the below descriptions your task is to draw a UML class diagram showing the various interfaces and classes and next simply state in sentences all the concepts that you learned from this exercise about inheritance, polymorphism and interfaces.
   1. An interface – Monster with a method menace() with return type as void
   2. An interface – DangerousMonster which inherits the Monster interface and has a method called destroy() with void return type.  
      (Note: Just like a class can inherit from another class, an interface can inherit from another interface using the “extends” keyword.
   3. An interface – Lethal with a method – Kill( ) with void return type.
   4. A class DragonZilla which implements DangerousMonster. This class does not have any methods of it’s own but provides implementation for the methods from DangerousMonster. The implementation of these methods simply prints meaning full statements that includes the name of the class and method using System.out.println() eg. “DragonZilla can be menace to children”
   5. An interface – Vampire that extends DangerousMonster and Lethal interfaces. This method has it’s own method – drinkBlood( ) with return type as void.  
      (Note: An interface can inherit from multiple interfaces but a class can inherit from at most one Class.)
   6. A class VeryBadVampire that implements Vampire interface. The methods that are inherited would simply provide an implementation that will print the name of the class and name of the method just like in step d.
   7. A class HorrorShow with the following implementation.

public class HorrorShow

{

public static void u(Monster b)

{

b.menace();

}

public static void v(DangerousMonster d)

{

d.menace();

d.destroy();

}

public static void w(Lethal l)

{

l.kill();

}

public static void main(String[] args)

{

DangerousMonster barney = new DragonZilla();

u(barney);

v(barney);

Vampire vlad = new VeryBadVampire();

u(vlad);

v(vlad);

w(vlad);

}

}

(30 points)

**Things to turn in:**

* For Exercise 1, Copy the source code for all the classes including your client program along and with contents of output window for two runs of the program and paste them on your assignment document.
* For exercise 2, include the class diagram and answers to the question.