CSULB
CECS277

Lab

Use an inheritance hierarchy containing types of employees in a

Use an inheritance hierarchy to develop a company's payroll application. In this company, commission employees (who will be represented as objects of a superclass) are paid a percentage of their sales, while base-salaried commission employees (who will be represented as objects of a subclass) receive a base salary *plus* a percentage of their sales.

- Define class CommissionEmployee, which directly inherits from class Object and has as private instance variables a first name, last name, social security number, commission rate and gross (i.e., total) sales amount.
- Define a new BasePlusCommissionEmployee class that *extends* class CommissionEmployee (i.e., a BasePlusCommissionEmployee *is a* CommissionEmployee who also has a base salary).

show how the BasePlusCommissionEmployee subclass can use CommissionEmployee's public methods to manipulate (in a controlled manner) the private instance variables inherited from CommissionEmployee.

For each class, you should define the:

- **equals method:** This method compares two objects for equality and returns true if they're equal and false otherwise. The method takes any Object as an argument. When objects of a particular class must be compared for equality, the class should override method equals to compare the contents of the two objects.
- **getClass:** Every object in Java knows its own type at execution time. Method getClass returns an object of class Class (package java.lang) that contains information about the object's type, such as its class name (returned by Class method getName).
- **hashCode:** This method provides the **hash code of an object**. Basically the default implementation of hashCode() provided by Object is derived by mapping the memory address to an integer value. Override the hashCode method in your implementation class, so if two objects are identical, then their hashCode value should be the same
- toString method that returns the class of the object and a String representation of the object.