The purpose of this lab is to get practice at using inheritance.

**Part 1 – Set up a class hierarchy for an employee system.**

**Note**: As per last week, you can use the same java project and package within the project as you did last week – or set up a *new* package within the same project. Or set up a new project and package. It’s up to you.

**Employee**

Create a new class called **Employee**. It needs attributes firstName (String), surName (String), staffNumber (int) , baseAnnualSalary (double) and startDate (String).

Include a Contructor that takes in all attribute values when an object of Employee is created.

Include a method – calculatePay() – which returns the monthly pay i.e. the baseAnnualSalary divided by 12.

As always, encapsulate the attributes and put in getters and setters (get help on this under the Refactor menu as demo’d in class!

**Hourly Employee**

An hourly employee is someone who is paid only for the hours that they work each week. So, their salary is calculated differently to employees, and they have an hourly rate. Their annual salary is always set as zero.

Create a new class called **HourlyEmployee** that inherits from Employee. It has extra attributes of *hoursWorked* and *hourlyRate* – both of type double.

How many attributes does an object of type of HourlyEmployee class have?

Include a Contructor to set up all the attributes of HourlyEmployee. Remember to call the constructor of the superclass, Employee.. when you are doing this as this already has done the work of setting the attributes that are part of the Employee class .

Include a method – calculatePay() – which returns the pay as hours worked multipled by the hourly rate. This method is **overriding** the employee method.

**Employee on Commission**

An employee on commission is someone who has a low base annual salary, and whose salary is then topped up by commission earned on sales.

Create a CommissionEmployee which inherits from the Employee class. It has a specific attirbute of “commissionEarned” for the month.

How many attributes does an object of type of CommissionEmployee class have?

Include a Contructor to set up all the attributes of HourlyEmployee .

Include a method – calculatePay() – which returns pay calculated as base annual salary divided by 12, plus the commissionEarned. This method will be **overriding** the employee method.

**Part 2 – Test it out.**

When you have set up your classes, set up the usual separate control class (Calling it Control as the class name is a good idea to make its purpose clear) – and put a main method in it.

Instantiate one object for each of your three class types: Employee, HourlyEmployee, CommissionEmployee.

You won’t see anything as there’s nothing being outputted – your three objects are just sitting there in memory, created.

Add a toString method to each of your classes. Each one should return a String.. consisting of the attributes of the object in the usual way: “This object has a firstName of .. and a surname of . etc…”. Make sure to use the super keyword from your subclasses to avoid rewriting code in the subclass toString() methods.

**Part 3 – Polymorphism**

In this part, you’ll implement code to demo polymorphism: The right behaviour happens, based on dynamic checking of the object type at run time.

Create an array of Employees (Employees[] myEmployees = Employees[6])

Instantiate each of the 10 entries with a variety of employees, hourly employees and commissionemployees..

e.g. myEmployee[0] = new … whatever class object you are creating..

Then put in a loop to call the Employees, from entry 0 to entry 6 – as shown in class:

employees[i].calculatePay()

The right version of calculatePay() will get called, without you having to figure out which object it’s in. Dynamic binding. Polymorphism. Yayy.