**Assignment 13 – Interfaces & Inner Classes**

The Date class was made an inner class of the employee class, placed at the beginning per the text’s recommendation. Date was originally made static to make this work, but upon reading a bit more it seemed that ‘nested static classes’ are not the same as inner classes. I created an instance of Employee to allow for the creation of the Date object I wanted to pass into the HourlyEmployee object I was creating. Creating this Employee object solely for that purpose didn’t seem great, so I set the Name and HireDate parameters using the Employee class’ mutators and made a new constructor in HourlyEmployee to accept the original employee object into its super(), along with the wage and hours. Also, the default employee constructor’s value of ‘Jan’ in creating the Date variable threw an error and needed to be corrected to ‘January’ in order to satisfy the switch, and I added the word ‘hired’ to HourlyEmployee.toString().

I *think* if I were using these classes, I could work with the Date as static without a problem, but I didn’t want to get caught on a technicality, using a ‘nested static class’ instead of a true inner class. If I left it, I could just create that date object with Employee.Date ‘datename’ = new Employee.Date instead of how it’s currently done.

