**CIS163 Lab**

**Polymorphism, an Inheritance Lab**

**Preparation:**

Bring your book to lab.

Attended class and stayed up to date with class material

**Objectives** (after completing the lab you will be able to do:)

* Apply polymorphism, an inheritance concepts to your code.

**Activities (steps):**

**(PLEASE COMPLETE EACH STEP COMPLETELY BEFORE MOVING TO THE NEXT STEP.)**

1. ***Task’*** Download the book’s code found on BB (LabSimplePoly.zip) and execute the Firm class. That is, get to know the code by using the debugger, execute the Firm class main method (contained in LabSimplePoly.zip) and use the step into and step over functions to watch how the program executes, i.e., step by step through the code. Does it work as expected? If the execution does not follow the expected path, ask your instructor why.
2. ***Task’*** Using the debugger (and use the step into and step over functions), how may instruction are executed when you instantiate a new Staff() at line 15 in the Firm class. MOST IMPORTANT, can you predict were is will go next. If you, you have completed this step.

Total number of instructions: \_\_\_\_2+42+40+20+75\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Between 30 and 90.

1. ***Task’*** Remove the abstract pay method from the Staffmember class, and replace it with:

public double pay() {  
 return 0.0;  
}

***Task’*** Does the output change? Yes or No If yes, how and why?

No change to the output.

***Task’*** What is the overall effect of this change in the Firm class (LabSimplePoly) program?

This change simply creates another write over point and on the way to the payment and also ultimately, this has no change as the method will need to be called on in the class

1. ***Task’* C**reate a SuperExecutive class that extends Executive. Add a new property (instance variable) named:
   1. protected double extraBonus;

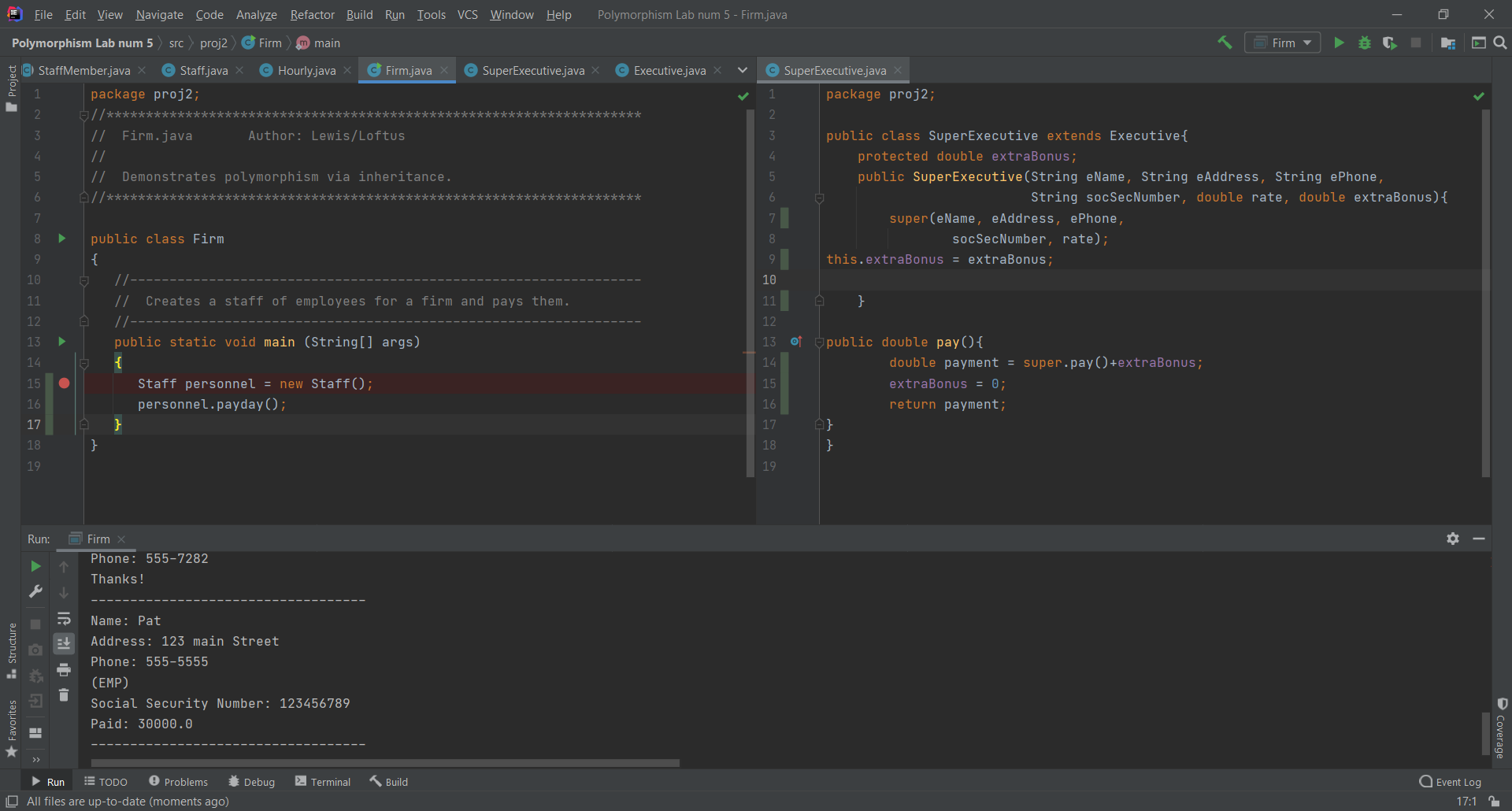
Create a pay method that first calls the pay method found in Executive and then adds on this extraBonus; (Hint; examine the pay method in the Executive class and see how that calls the super pay method).

Create a constructor for SuperExecutive that accepts 6 parameters; first call the constructor for Execute using super and the first 5 parameters, then assign the last parameter to extraBonus. (Hint; examine the constructor in the Executive class and see how that calls the super method).

Add an SuperExecutive to the staffList array found in the Staff class constructor as the 7th person. See the code below for adding a 7th person in the staffist array. Be sure there is enough room in the array (6 to 7) to hold this superExecutive object.

staffList[6] = new SuperExecutive ("Pat", "123 main Street","555-5555", “123456789”, 10000.0, 20000);

1. ***Task’*** Using the debugger, execute the program and use the step into function to watch how the program executes, i.e., step by step through the code. Does it work as expected for SuperExecutive? What is the output for SuperExecutive? If the execution does not follow the expected path, ask your instructor.

Yes, the output is 

1. ***Task’*** Change the code, that is, rewrite the code to allow the removal of the cast statements (see below) from the Staff class constructor method.

~~((Executive)~~staffList[0]).awardBonus (500.00);

~~((Hourly)~~staffList[3]).addHours (40);

How did you accomplish this?

1. ***Task’*** Change the code in the Executive and Hourly class by removing the pay Method. What are the effects of doing this? Does it still compile or does this cause errors? Execute the program Firm and step through the code with the debugger. Does the execution path change, and if so, how?

Compiles, but since not overridden, but it is something that has affects

If you are done with this lab, start working on Project 2.