**CIS163 Lab 6 (Poly)**

**Polymorphism, interfaces, 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, interfaces, an inheritance concepts to provided code.

**Activities (steps):**

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

1. **Download the book’s code found on BB (LabPoly.zip) and execute the code. Examine the output and take a few minutes to understand the output; validate that the output is correct.**
2. **Using the debugger, execute the program (contained in LabPoly.zip) 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? If the execution does not follow the expected path, ask your instructor why.**
3. **Create an interface with one abstract method**

**public** **interface** PayInterface {

**public** **abstract** **double** pay();

}

1. **Remove the abstract method from the Staffmember class, and instead, implement the PayInterface in the StaffMember class.**
   * 1. **Use the keyword implements to implement PayInterface, not extends.**
     2. **What changes are needed in the Staffmember class?**
     3. **Is the following statement true or false?**
        + 1. **The Employee class must implement the pay method still.**
2. **Examine the output and take a few minutes to understand the output; validate that the output is correct. Does it work as expected?**
3. **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? If the execution does not follow the expected path, ask your instructor.**
4. **In this step, change the code, that is, rewrite the code (page 419) that would allow you to remove the cast statements (see below) from the staff class constructor method.**

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

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

1. **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? If the execution does not follow the expected path ask your instructor.**

(Once you accomplish this step, please put it back to where the cast is needed)

1. **Now, create a SuperExecutive class that extends Executive. 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.**

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

(also, create a default constructor, see how that would work too).

1. **Be sure to create the pay and toString methods inside the SuperExecutive class.**
2. **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? If the execution does not follow the expected path, ask your instructor.**
3. **Change the code in the Executive and Employee class by removing the pay Method. What are the effects of doing this?**

1. **Examine the output and take a few minutes to understand the output; validate that the output is correct. Does it work as expected? What has changed?**
2. **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? If the execution does not follow expected the path, ask your instructor.**
3. **Try to put the code back into place that requires the two cast statements. Then, change the code in the Staff constructor, specifically, the two cast statement lines:**

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

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

**so that the lines 19 – 33 can be in any order. In other words, you will not know stafflist[0] is an executive and staffList[3] is an hourly. They could be in any order.**

1. **Finally, for this lab, if you have time, goto page 261 and create a UML diagram for that program, named, StyleOptions.**