# LAB\_03 -- Inheritance, Polymorphism, and Interfaces

# Summary

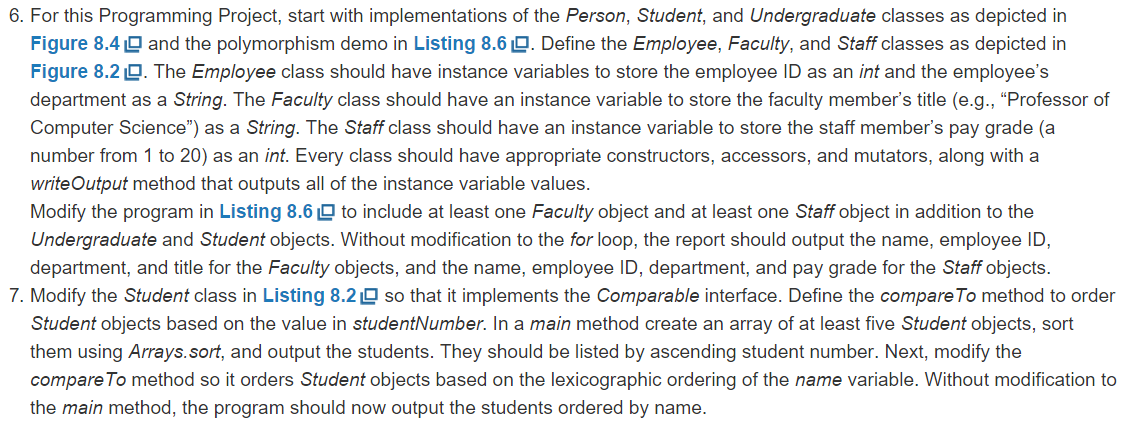
This lab provides practice for several object-orientated design concepts – Inheritance, Polymorphism, and Interfaces.

# Acknowledgement

The lab has been motivated by Exercise #6 and #7 in the book *Java: An Introduction to Problem Solving and Programming, Seventh Edition* by Walter Savitch

# Lab

Complete labs #6 and #7 from the text on pg 664, as shown below. I have included the Figures and Listings from the text at the end of this assignment for convenience.



# Additional Requirements for #6

1. Modify the person and employee classes so that they may not be instantiated directly – meaning they can contain logic but only child classes can be created.
2. The following preconditions must be implemented. Alert the user and abort the program if an error is encountered.
   1. A student’s student number must be positive
   2. An undergraduate’s level must be between 1 and 4, inclusive
   3. An employee’s id must be positive
   4. A staff member’s paygrade must be between 1 and 20, inclusive
3. Implement an equals(Faculty otherFaculty) and equals(Staff otherStaff) for the Faculty and Staff classes respectively. Your methods must rely on logic inherited from the Employee base class.
   1. Faculty members are equal if they share the same employeeId. Only employeeId is considered.
   2. Staff members are equal if they share the same employeeId . Only employeeId is considered.
4. Include simple tests in your demo class or unit tests for the following items:
   1. Test your Faculty.Equals(…) method
      1. Include an assertion to verify a known true case
      2. Include an assertion to verify a known false case
   2. Test the pay grade precondition works as expected.
      1. Include an assertion to verify a known true case – between 1 & 20 inclusive
      2. Include assertions to verify a bad cases

# Additional Requirements for #7

1. Keep all versions of your comparable methods. Comment out the prior versions, so I can review the work.
2. After getting the Array.sort(…) to work via the Comparable interface, implement your own sorting method using the Bubble sort algorithm we just learned. Your Bubble sort should sort the Student object’s in ascending order by Student Id. Feel free to do this work in the class which contains your main method.

# Figures and Listings

Refer to the following figures from the text while working on the lab:

