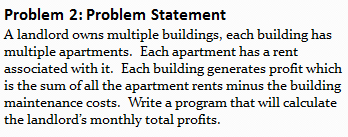
**Lab 3**

1. In the code folder for this lab there is a package lesson3.labs.prob1 containing two classes, Person and PersonWithJob. In each class, the equals method has been overridden. Run the main method in the PersonWithJob class. In the main method, two instances of Person have been compared to determine if they are equal. The comparison is done in two different ways. One way leads to a “false”, the other to a “true.” Explain why this has happened. Then provide a solution by replacing inheritance with composition.

ANSWER: This happened because in the first case, the comparison is done using the equals() method in the PersonWithJob class which returns false because Person is not an instance of PersonWithJob. Whereas, in the second case, the comparison is done using the equals() method in the Person class which returns true because PersonWithJob is an instance of Person. This is called the Asymmetry problem (violates contract for equality).

1. Design a solution to the problem given below, and then implement in code. A Main class has been provided in the startup code that hard-codes values and attempts to test your code. You may wish to modify the Main class so that it matches your implementation of the other classes – it is provided as a convenience. The main method should calculate the income for the hardcoded values. (Note that the problem in this case is different from the one given in Lab 1.)



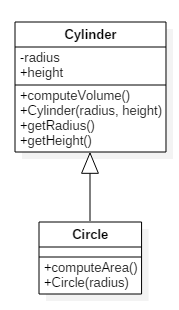
1. A. UML classes Circle and Cylinder are given below, pictured in an inheritance relationship. Write

the code for Circle and Cylinder in Java, making use of the inheritance relationship. Does it make

sense to use inheritance here? Explain.

ANSWER: It does not make sense because the CIRCLE class inherits some instance variables which it does not use (height).

B. Redo the design using composition, and write the resulting code in Java.



|  |
| --- |
|  |

1. In Lab 1 a properties management system was introduced. In that lab, you specified classes and some attributes for those classes. In this exercise, think of a way to design further using inheritance and include associations (with multiplicities) and some operations for you classes. Then translate your diagram into Java code. Both an Admin and Driver class have been provided in your code folder. The Driver class creates some instances of the different properties and passes these into the Admin method computeTotalRent; this method performs a correct computation, but the implementation proceeds by checking the types of the different rental properties. Refactor the implementation of computeTotalRent so that the inheritance you have introduced is used, together with polymorphism. Below is provided the problem statement and a solution for the Lab 1 exercise.

