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.

The first comparison leads to false because the comparison goes to the ‘equals’ method is class ‘PersonWithJob’. The class tries to compare the other object ‘aPerson’ which is an instance of Person (ie superclass of PersonWithJob) to PersonWithJob, which fails and returns early. The class does not have the necessary salary attribute that the comparison in class PersonWithJob considers, which is why it fails.

The second comparison is done with p2 against p1 which goes to the method ‘equals’ in class Person. The method check ‘aPerson’ is an instance of Person, which is true since PersonWithJob is a subclass of Person and hence the instance of PersonWithJob is an instance of Person. The class just checks the equality of names in the cases which is evaluated as true.

2. 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.)

The code for this solution is uploaded as a separate zipped file.

3. 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.

Although circle is a type of cylinder, it is not a perfect cylinder since it is a 2 dimensional figure with 0 height. The Cylinder has getHeight and computeVolume methods according to the UML diagram presented, which are not relevant in case of circle. So, it does not make much sense to use inheritance here.

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

The code is uploaded as a zipped file.

4. . 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.

Diagram

Description automatically generated

Code for this exercise is uploaded as a part of the zipped file uploaded in the same compressed zipped file.