

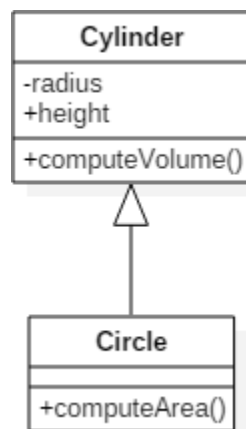
## Lab 3

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.
  - *PersonWithJob is a Person, but the opposite is negative. Therefore we cannot compare a superclass with a derived class.*
2. Design a solution to the problem given below, and then implement in code. Hard-code a few buildings, apartments and their rental fees in the main method of a separate `Main` class to test your code. In your main method, calculate the income for your hardcoded values. (Note that the problem in this case is different from the one given in Lab 1.)

### Problem 2: Problem Statement

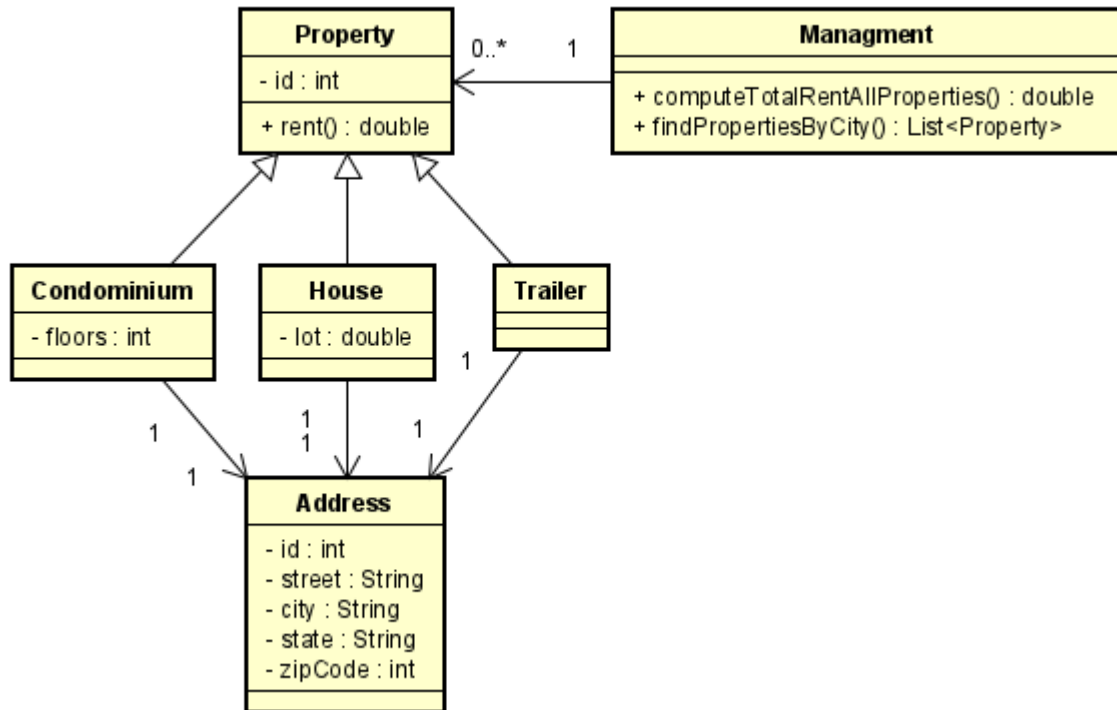
A landlord owns multiple buildings, each building has multiple apartments. Each apartment has a rent associated with it. Each building generates profit which is the sum of all the apartment rents minus the building maintenance costs. Write a program that will calculate the landlord's monthly total profits.

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.
  - No, Even mathematically been explained that a circle is a cylinder with a height of 0, this implementation could break the code if a circle receive a height greater than 0.
- B. Redo the design using composition, and write the resulting code in Java.



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 your 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.

**Problem Description:**

A landlord owns several types of properties: houses, condominiums, and trailers.

A house has an address and a lot size. Rent for a house is computed by

$$\text{rent} = 0.1 * \text{lot size}$$

A condominium has an address and a certain number of floors (1 floor, 2 floors, or 3 floors). Rent for a condominium is computed by

$$\text{rent} = 400 * \text{number of floors}$$

A trailer belongs to a particular trailer park (specified by the trailer park address).

The rent for a trailer is always \$500.

The property management software is required to have an Admin module that supports various functions. One of these functions is to compute total rent for all the properties registered in the system. Another function is to list all properties in the system that are in a specified city.

