Task 1:

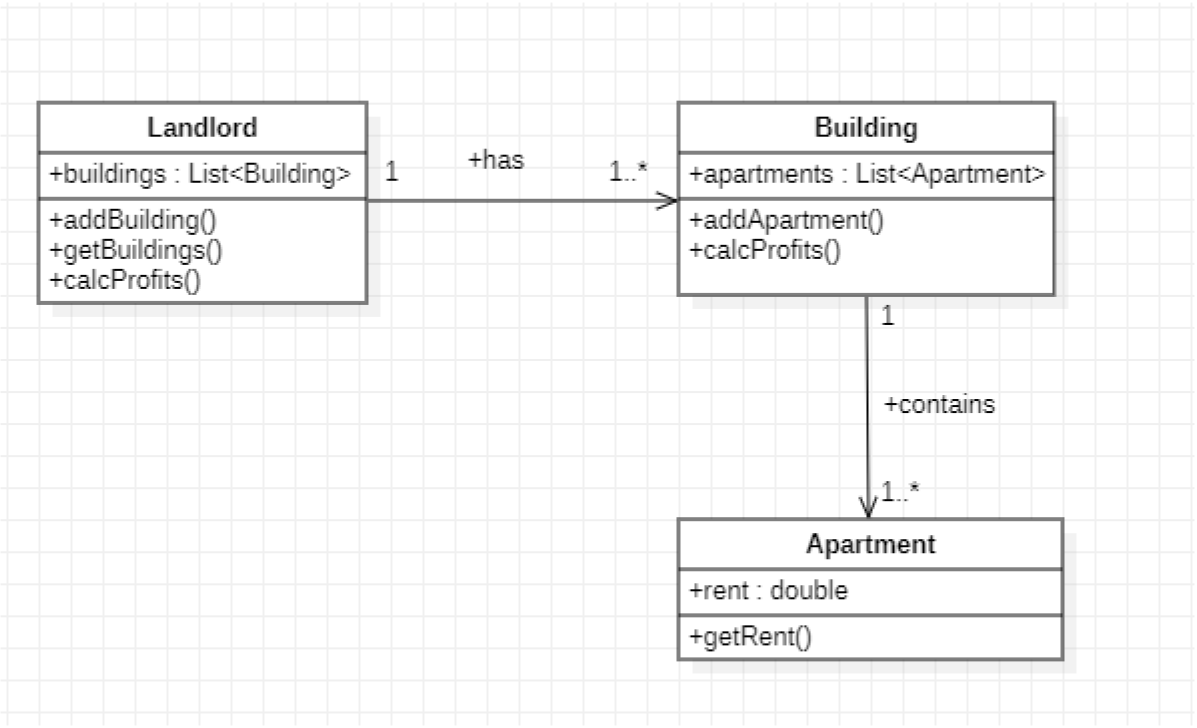
In the inheritance code, an instance of PersonWithJob is created which is p1 and an instance of Person is created which is p2.

First case: When a compare is done with the equal method of p1 (PersonWithJob) then equal method returns false because p2 (Person) is not type of PersonWithJob.

Second case: When a compare is done with the equal method of p2 (Person) then equal method returns true because p1 (PersonWithJob) is also type of Person.

This is solved by using composition instead of inheritance. Solution is in project file. Lab3.Problem1.

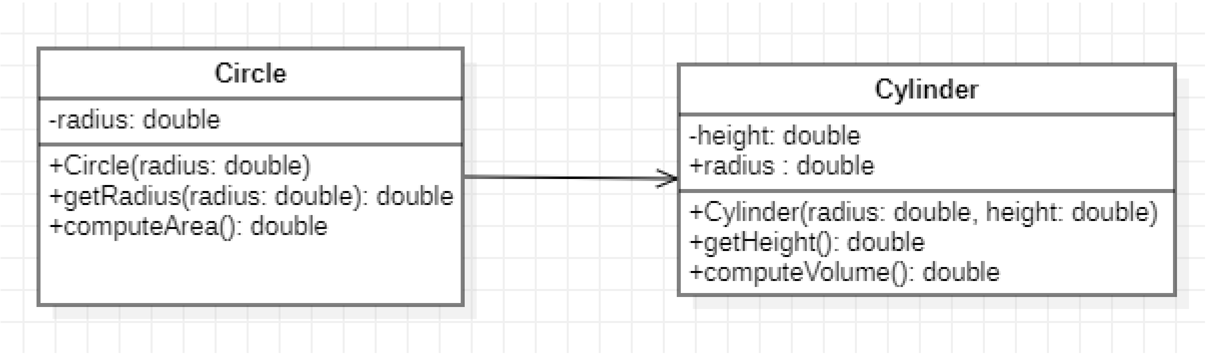
Task 2: (Coding implementation is in project folder.)



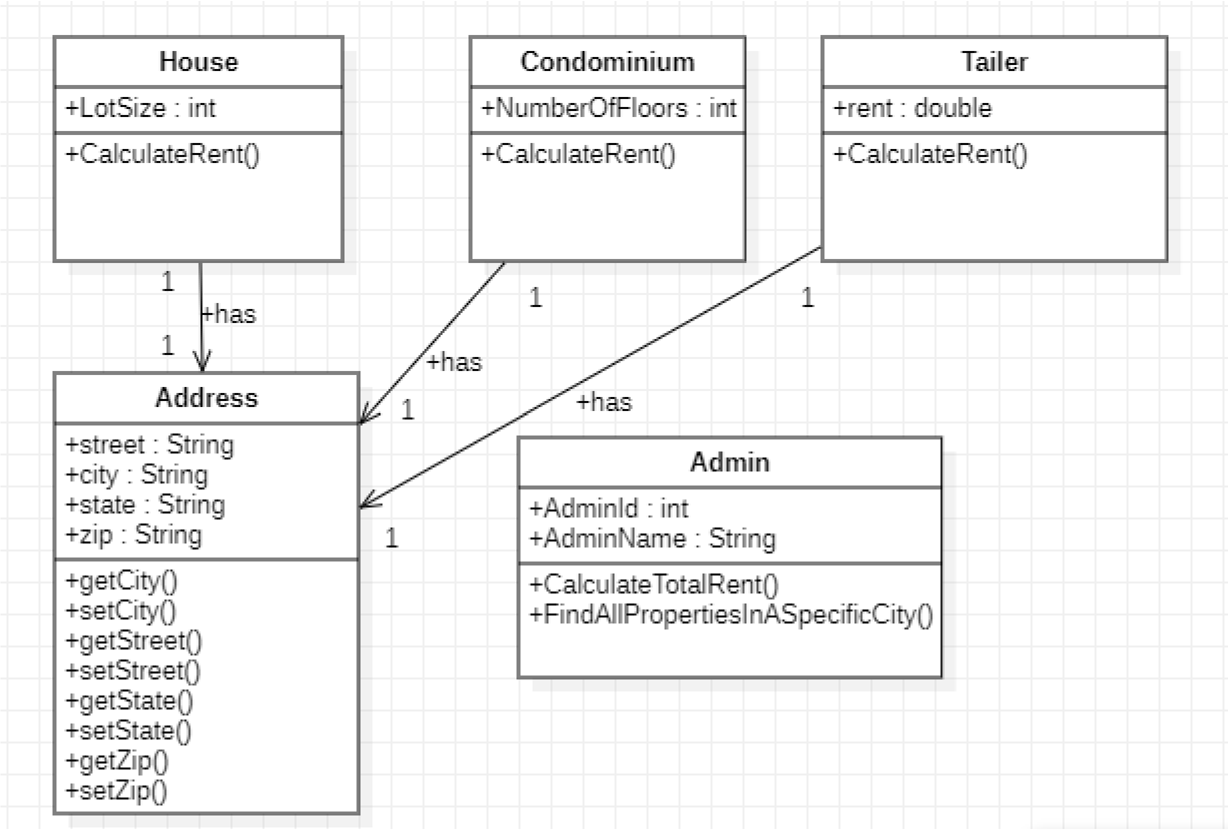
Task 3: (Coding implementation is in project folder.)

If we use inheritance, it doesn’t make any sense in this case. Suppose if we create Cylinder cy = new Circle() then we can only access the computeVolume method. It can’t access the computeArea method. So according to LSP it doesn’t make sense to use inheritance.

Design for composition:



Task 4: (Coding implementation is in project folder.)


House
+LotSize : int
+CalculateRent()
Condominium
+NumberOfFloors : int
+CalculateRent()
Tailer
+rent : double
+CalculateRent()
1
+has
1
1
+has
1
1
+has
1
Address
+street : String
+city : String
+state : String
+zip : String
+getCity()
+setCity()
+getStreet()
+setStreet()
+getState()
+setState()
+getZip()
+setZip()
Admin
+AdminId : int
+AdminName : String
+CalculateTotalRent()
+FindAllPropertiesInASpecificCity()