**Question 1**

Two statements in the code return false and true because of late binding(method overriding) feature of inheritance.

PersonWithJob.equals method is checking an extra property “salary” in its equals method, and comparing instance of PersonWithJob with instance of Person, is unsatisfied condition for a person because all PersonWithJob objects are Person objects(upcasting), but all Person objects are not PersonWithJob objects(downcasting).

We can prevent this kind of situation using composition, where we make equals method of Person class un-overridable using final keyword so that same equals method from Person class is called every time when we call it from either of the objects of Person or PersonWithJob classes.

**Code : modified code files have been uploaded with this file.**

**Question 2**

Diagram

Description automatically generated

**Code : code files have been uploaded with this file.**

**Question 3**

Circle and cylinder are not compatible i.e. cylinder is not of type circle, so there is no meaning of inheriting cylinder by circle. Rather they are both shapes and can inherit Shape class which will have radius, computeArea in common. To use composition, we will not inherit the circle class, rather we create a circle class inside Cylinder class and use the circle computerArea method to compute volume of cylinder.

**Code : code files have been uploaded with this file.**

**Question 4**

Diagram

Description automatically generated

We can re-develop this problem using inheritance and polymorphism using abstract class, method overriding and upcasting.

**Code : code files have been uploaded with this file.**