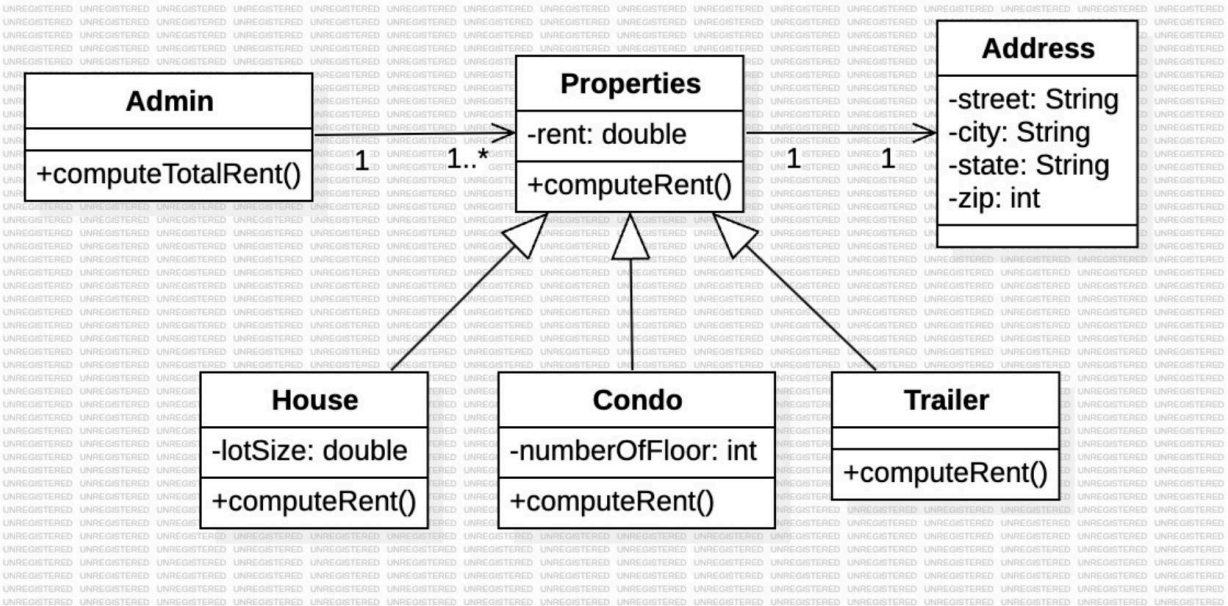
Lab 3

1. **Two objects equals function checking Object type.**

   The first one printed false, because the first equal method checks against two PersonWithJob classes. Parent Person class can't be downcasted to PersonWithJob class which is a child class.
   PersonWithJob => if(!(aPersonWithJob instanceof PersonWithJob)) return false;

   The second returned true, because p1 is PersonWithJob and p2 is Person when downcasted can be Persons and they have the same name. Because a child can be downcasted to parent class.
   Person=> if(!(aPerson instanceof Person)) return false; //This condition passed which means it doesn't return false

2. **Class Diagram**



3. **Does Inheritance Make Sense Here?**

Pros:

- **Code Reusability**: The `computeArea()` method from `Circle` is reused in `Cylinder`.

- **Logical Relationship**: Cylinders have circular bases, so there is a conceptual "is-a" relationship.

Cons:

- **Incorrect Semantics**: A `Cylinder` is not a type of `Circle`. It comprises one or more circular bases and an additional dimension (height). Inheritance here needs to represent this relationship properly.
- **Reduced Flexibility**: If the `Cylinder` class needs to support non-circular bases in the future, the inheritance structure becomes limiting.
- **Violation of Liskov Substitution Principle**: A `Cylinder` cannot always replace a `Circle`, as some operations on a `Circle` (e.g., scaling the radius) might not be meaningful for a `Cylinder`.