```java
package assignment3.problem3.inheritance;

public class Cylinder {

    private double height;

    private double radius;


    public Cylinder(double radius, double height) {
        this.height = height;
        this.radius = radius;
    }

    public double getHeight() {
        return height;
    }

    public double computeVolume() {
        return Math.PI * Math.pow(radius, 2) * height;
    }
}
```


Circle

```java
public class Circle extends Cylinder {

    private double radius;

    public Circle(double radius, double height) {
        super(radius, height);
        this.radius = radius;
    }


    public double computeArea() {
        return Math.PI * Math.pow(radius, 2);
    }
}
```


Question: 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.

Explanation :
Incorrect Use of Inheritance

The relationship between `Circle` and `Cylinder` does not follow SOLID principles:

- A **circle is not a cylinder**; they represent different shapes.
- A `Circle` should not extend `Cylinder` because inheritance represents an **is-a** relationship, and a circle is a 2D shape, while a cylinder is a 3D shape.
- Inheritance here violates the **Liskov Substitution Principle** (LSP). We can't substitute a `Circle in place of a Cylinder object without the code breaking(logically).` A `Circle` cannot logically substitute a `Cylinder` because it does not share the same characteristics (e.g., height is meaningless for a `Circle`).