

## The University of the West Indies, St. Augustine COMP 3607 Object Oriented Programming II 2020/2021 Semester 1 Lab Tutorial #4

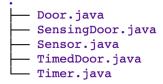
This tutorial focuses on the SOLID design principles.

## **Learning Objectives:**

- Reinforce your knowledge on each of the SOLID design principles
- Discuss examples of SOLID design principles in code snippets
- Refactor existing code to conform to one or more SOLID design principles

## **Section A: Interface Segregation Principle**

The following files are used in this section. Download the java source and examine the code.



- 1. What is the purpose of timeOutCallback() method of the Door interface?
- 2. What is the purpose of proximityCallback() method of the Door interface?
- 3. What method(s) of the Door interface is TimedDoor class forced to implement, even though it doesn't need it?
- 4. What method(s) of the Door interface is SensingDoor class forced to implement, even though it doesn't need it?

5. How does this design violate Interface Segregation Property?
6. How would you change the design so that ISP is not violated?
7. Draw a class diagram to capture these design changes.
8. Refactor the code to reflect your design changes in an individual project repository.
Section B: Dependency Inversion Property
The following files are used in this section. Download the java source and examine the code.
Button.java Lamp.java
There are two classes defined here: Button and Lamp. Are there any dependencies between them?     Which class depends on the other?
<ol> <li>Suppose we wanted to add a Fan class that can be turned on and off to this design and a Fan also needed a button.</li> <li>(a) Describe the problems that may occur.</li> </ol>

	(b) Suppose we handle this by adding a FanButton class that allows the Fan to be turned on and off. What would the class diagram look like in this case (including Lamp, Button, Fan, and FanButton)? Draw the diagram.
3. \	What is the common behaviour between Lamp and Fan?
á	Suppose we define an Equipment interface for the common behaviour of Lamp and Fan classes and have Lamp and Fan classes implement that interface. What would the class diagram look like in this case?
	We can now modify the Button class to have a reference to an instance of Equipment object.  Draw the class diagram after this modification:
6.	Refactor the code to reflect your final design in an individual project repository.

## **Section C: Multiple Principles**

- 1. Explain how the Open/Closed and the Liskov Substitution principles are automatically satisfied when the Dependency Inversion principle is adhered to.
- 2. Consider the code snippet below in relation to the Single Responsibility (SR) Principle

```
public class Customer{
    //state variables and constructors
    public void storeOrder(Order o){ ... }
    public Order findOrder(int orderID){ ... }
    public boolean cancelOrder(Order o){ ... }
    public String getCustomerName(){ ... }
    public String getCustomerAddress(){ ... }
    public String getCustomerEmail(){ ... }
}
```

Figure 1

Assuming that there are no errors in the code in Figure 1:

- (a) Explain how the SR principle is being violated.
- (b) Discuss how you would refactor the code so that the SR principle is followed.
- (c) Identify ONE substantial benefit of following the SR principle.
- (d) Suggest a simple technique that a programmer can use to avoid violating the SR principle when adding a new feature to an existing class.