Laboratory Report

# Structural Design Patterns: Adapter & Bridge

|  |  |  |
| --- | --- | --- |
|  | | |
| Student ID: | 3684104 | |  |
| Student Name: | | Cooper Dickson | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Course: |  | CS4015 |  | CS 6075 |  | SWE 4403 |

## Introduction:

Write a brief introduction to you report.

## Answers to exercises

1. You are designing an application that has two classes: Person and Customer. Person uses *fullname* information, but Client wants to use lastName, firstName as Customer information. In this exercise, you should use an Adapter pattern to get Customer information from a list of persons.
   1. Create a UML diagram that uses an adapter to test how the client class can get customers from a list of persons.

Diagram

Description automatically generated

* 1. Write the code and run it. Your output should list the persons using lastName, firstName format. Hint: A possible Java implementation of the Person class is shown below:

**package com.designpatterns.adapter;  
  
public class Person {  
 private String fullName;  
  
 public Person(String fullName) {  
 this.fullName = fullName;  
 }  
  
 public String getFullName() {  
 return fullName;  
 }  
  
 public void setFullName(String fullName) {  
 this.fullName = fullName;  
 }  
  
 @Override  
 public String toString() {  
 return "Person [fullName=" + fullName + "]";  
 }  
}**

Insert your code or link to your code here.

Insert the console output here

1. A team of developers is considering the following structures to create an application that simulates production and assembling of vehicles.
   1. Which diagram represents the best solution in terms of extendibility and reusability? Justify your selection.

|  |
| --- |
| **Solution a**  Vehicle  Bus  Bike  AssembleBike  ProduceBike  AssembleBus  ProduceBus |
| **Solution b** |

Select your solution below:

Solution a

Solution b

Justification:

The solution b is the better solution because it keeps the classes separate and allows a developer to create new implementations independently from each other. This design also allows for the classes to follow the single responsibility principle by keeping classes concise and allowing for similar methods to be reused, instead of having to implement extremely specific methods for each subclass. The reusability id solution b is better too because new types of Vehicles can be added without having to add in two subclasses to define the Assemble and Produce methods.

* 1. The file in [L04Exercise02.zip](https://lms.unb.ca/d2l/le/content/217659/viewContent/2453461/View) contains the code for one of the previous solutions.
     1. Do you recognize (or would suggest) the use of any design pattern?

This UML represents the bridge pattern. This is because rgather then structire the cide as a big hierarchy, it is divided so that the workshop methods are indepedndant of the vehicle subclasses. The vehicle class can then use the Assemble and Produce class instead of making new classes, representing thebridge connection.

* + 1. List and identify the components of the design pattern identified or suggested.

Implementation – Workshop

Concrete Implementation – Produce, Assemble

Abstracation – Vehicle

Redefined Abstraction – Bike, Car

Client – Demo

* + 1. Add a new vehicle (Truck) to your application.

Show your new code here or provide a link to your source code.

* + 1. Run the code and copy the output.

Show the console output here. A screenshot is preferable.

## Conclusions

Provide brief conclusions of your lab exercise. You may also add suggestions on how to improve the lab exercise for future students.