Laboratory Report

# Structural Design Patterns: Proxy, Flyweight, facade

|  |  |  |
| --- | --- | --- |
|  | | |
| Student ID: |  | |  |
| Student Name: | |  | | |

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

## Introduction:

Write a brief introduction to you report.

## Answers to exercises

We are going to build an [Object-Relational Mapper](https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping) (ORM). ORMs are tools responsible for mapping data between a relational database and an object-oriented model. There are many popular O/RMs out there such as [Hibernate and Entity Framework](https://en.wikipedia.org/wiki/List_of_object%E2%80%93relational_mapping_software).

The file [Lab06Exercise01.zip](https://lms.unb.ca/d2l/le/content/217659/viewContent/2459867/View) contains a simplified implementation of an imaginary ORM.

The DbContext class is used to read or write data to the database.

The Demo class shows the typical workflow for using a DbContext

* + Read an object from a database (dbContext.getProduct())
  + Change the properties of the in-memory object (product.setName())
  + Ask the dbContext to save the changes (dbContext.saveChanges())

1. Run the Demo class. You should see a SELECT statement written on the terminal. This simulates reading a product record from a database. Please show the output of your run here.
2. What is missing is the two UPDATE statements that should be generated when we save the changes. The reason this is happening is that the DbContext cannot keep track of the changed objects. So, when the call to saveChanges() is made, nothing happens.

This problem can be solved using the proxy pattern. A proxy object looks like our target object (eg a Product object) but it adds some extra behavior to it. Let’s call it ProductProxy which inherits from Product as:

**public class ProductProxy extends Product {  
 private DbContext context;  
  
 public ProductProxy(int id, DbContext context) {  
 // put your code here.**

**}  
  
 @Override  
 public void setName(String name) {  
 // put your code here.**

**}  
}**

When we call the setName() method, the proxy should notify the DbContext that it has changed. DbContext provides a method for this: markAsChanged(). Use the proxy pattern to allow DbContext keep track of changed objects and persist them in a database.

* 1. Complete the code of the ProductProxy class.
  2. Run the demo class and print the output

NOTE: Some real-world ORMs automatically generate these proxy objects based for you, so you don’t need to code them by hand.

### Exercise 2

In this exercise two versions of a spreadsheet application like Excel or Numbers are built.

1. Check the classes in the [Lab06Exercise02a.zip](https://lms.unb.ca/d2l/le/content/217659/viewContent/2459868/View) file and see the implementation of Cell and SpreadSheet concepts.
   1. Run the code in the Demo class and show your outputs here.
   2. Draw the UML structure of the code.
2. Decompress and run the code in [Lab06Exercise02b.zip](https://lms.unb.ca/d2l/le/content/217659/viewContent/2459869/View) which implements a flyweight pattern. Is there is any difference with the output previously seen?
   1. Check the implementations of Cell, CellContext, and CellContextFactory. Draw the UML structure of the code.
   2. Explain under which scenarios this implementation is better than the previous one?

### Exercise 3

You are building a social media management tool. On a few screens, we need to display the recent tweets of a given user. The code is presented in the file [L06Exercise03.zip](https://lms.unb.ca/d2l/le/content/217659/viewContent/2459870/View). Look at the code in the Demo class. This class shows the steps required to talk to the Twitter API.

* + You need to get a request token first.
  + You will then exchange the request token with an access token.
  + To get the recent tweets, you need to send the access token to Twitter API.

We have to repeat similar steps for other operations such as composing a new tweet, liking a tweet, etc.

1. What are the problems with the current implementation?
2. Use the facade pattern to solve these problems and modify the Demo class to use the façade designed.
3. An incident management organization has an application that allows it to deploy available teams when an incident (e.g., fire) occurs. A team often includes a truck and one or two persons, but sometimes it may include various sub teams like in the following example:

* Team
  + Sub Team 1
    - Truck
    - Human Resource
    - Human Resource
  + Sub Team 2
    - Truck
    - Human Resource
    - Human resource

Look at the simulation of an implementation in the code provided by [Lab05Exercise01.zip](https://lms.unb.ca/d2l/le/content/217659/viewContent/2456637/View).

* 1. What are the problems that you see with this implementation?

Provide your answer here

* 1. Refactor the code using the **composite** pattern.

You may provide a link to your code or have a copy of it here

1. You are building a code editor like **IntelliJ** or **VSCode**. When we open a project, we should see all the artifacts in that project. Every artifact should have an icon and the icons can be different. The icons provide quick visual information to the user. For example, the files listed below by IntelliJ have different icons.

Graphical user interface, text, application

Description automatically generated





Similarly, if an artifact includes an error, its icon should include an error marker:



* 1. The compressed file [L05Exercise02.zip](https://lms.unb.ca/d2l/le/content/217659/viewContent/2456638/View) contains the code of the application. Note that the rendering of these icons is simulated using simple strings. Decompress the file and create a project with them.
     1. Run the code in the Demo class.

Show console ouput here

* 1. Refactor the solution using the decorator pattern.
     1. Run the refactored code.

Show console ouput here

* + 1. Upload the refactored code to D2L.

Show refactored code here or provide a link/information of its location.

## Conclusions

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