COS 214 Practical Assignment 2

- Date Issued: **25 August 2020**
- Date Due: **8 September 2020** at **8:00am**
- Submission Procedure: **Upload via Clickup**
- Submission Format: **archive (zip or tar.gz)**

1 Introduction

1.1 Objectives

In this practical you will:

- implement the State design pattern;
- implement the Strategy design pattern
- implement the Decorator design pattern
- implement the Composite design pattern;
- understand UML State Diagrams; and
- integrate the patterns.

1.2 Outcomes

When you have completed this practical you should:

- understand the State design pattern and how the state changes take place;
- understand how the Strategy pattern and how it differs from the Template Method;
- be able to tell the difference between the State and Strategy design patterns;
- understand how composite objects can be created and manipulated as a unit with the Composite design pattern;
- be able to add functionality to any object with the Decorator pattern; and
- be able to illustrate how functionality of a system can be adapted depending on the current state of the system.

2 Constraints

1. You must complete this assignment individually.
2. You may ask the Teaching Assistants for help but they will not be allowed to give you the solutions.
3. All diagrams need to be completed in Visual Paradigm. Failure to do so will result in 0 for the diagram.

3 Submission Instructions

You are required to upload all your source files (that is `.h` and `.cpp`), your Makefiles, UML diagrams as individual or a single PDF document, a text file labelled "readme" explaining how to run the program and any data files you may have created, in a single archive to Clickup before the deadline. Each section should be in a seperate folder and each section should have its own Makefile. Failure to follow any of these instructions may result in a mark penalty.

4 Mark Allocation

Task	**Marks**
Pandemic Class	5
The State Pattern	25
The Strategy Pattern	20
Section A Main	15
The Image Hierarchy	10
Decorator and Composite	20
Updating the Pandemic Class	10
Main	15
TOTAL	120

5 Assignment Instructions

In this assignment you will be creating a program to help the government model and demonstrate its response to the current pandemic and future pandemics.

Section A

Task 1: Pandemic Class .... (5 marks)

In this task you will create the very basic Pandemic class to demonstrate the use of the Strategy and State design patterns. You will expand this class in the following tasks.

1.1 • Private (5)

  – name - a string eg. "Covid-19"

Task 2: The State Pattern .... (25 marks)

In this task you will implement the State design pattern to illustrate the risk-adjusted strategy as the cases increase or decrease.(COVID-19 Risk Adjusted Strategy - SA Corona Virus Online Portal, 2020)

2.1 The LevelState Class. The LevelState class will fulfil the role of the State participant in the State design pattern, and should have the following members: (5)

- Private
  - pandemicLevel - a string. This will always be either "level1", "level2", "level3", "level4", "level5" or "none"
- Public
  - handle - this is an abstract method that does not have any parameters or a return value.
  - changePandemicLevel - this is an abstract method that has one boolean parameter(true -> increase level, false - > decrease level)and returns an instance of the LevelState class.
  - getPandemicState - this function returns the value stored in the pandemicLevel member variable for state identification during a run.

2.2 The Concrete LevelState Classes. You will implement the LevelOneState, LevelTwoState, LevelThreeState, LevelFourState, LevelFiveState and NoLevelState concrete state classes, which will inherit from the LevelState class. Each of these classes will have at least the following members: (10)

- Public
  - a constructor which takes no parameters and initialises the pandemicLevel member variable to the corresponding value stated in the previous section ("level1", "level2", "level3", "level4", "level5" or "none")
  - handle - this method must be implemented for all 6 concrete level classes and will print out the current level, as follows for each different class:

LevelOneState	"The country is at level1: Very few cases"
LevelTwoState	"The country is at level2: Take caution"
LevelThreeState	"The country is at level3:Moderate amount of cases, make sure you are wearing a mask"
LevelFourState	"The country is at level4: There are quite a bit of cases, take all necessary precautions."
LevelFiveState	"The country is at level5: Don't go outside unless you have to"
NoLevelState	"The country is at no level: Normal Living"

  - changePandemicLevel - this method must be implemented for all 6 concrete level classes and will return a new instance of the concrete LevelState which follows the current LevelState depending on the boolean parameter passed:

	True(increase)	False(decrease)
LevelOneState	LevelTwoState	NoLevelState
LevelTwoState	LevelThreeState	LevelOneState
LevelThreeState	LevelFourState	LevelTwoState
LevelFourState	LevelFiveState	LevelThreeState
LevelFiveState	LevelFiveState	LevelFourState
NoLevelState	LevelOneState	NoLevelState

2.3 Updating the Pandemic Class. The Pandemic class should be updated with the following: (10)

- Private
  - add the appropriate private member variable to the Pandemic class so it can fulfil the role of the Context participant in the State design pattern.
- Public
  - increaseLevel - This method will change the Pandemic LevelState to the next appropriate state and then call the handle method to print information about the new state. If it's impossible to change the state an appropriate message should be displayed.
  - decreaseLevel - This method will change the Pandemic LevelState to the previous appropriate state and then call the handle method to print information about the new state. If it's impossible to change the state an appropriate message should be displayed.

Task 3: The Strategy Pattern .... (20 marks)

In this task you will implement the Strategy design pattern to illustrate different pandemic response strategies.

3.1 The PandemicStrategy Hierarchy. The PandemicStrategy hierarchy will consist of 6 classes: (10)

- The PandemicStrategy class - This class represents the Strategy participant in the pattern, and will have an abstract takeAction() method, which takes no parameters and has a string return type.
- The levelOneStrategy, levelTwoStrategy, levelThreeStrategy, levelFourStrategy and levelFiveStrategy classes. These classes will represent the ConcreteStrategy participants in the pattern, and inherit from the PandemicStrategy class.
- The levelOneStrategy, levelTwoStrategy, levelThreeStrategy ,levelFourStrategy and levelFiveStrategy classes will implement the takeAction() method to return a string unique to each class. Keep in mind that these strings will be combined the pandemics name, so only partial sentences will be returned here:
  - levelOneStrategy - " has started to spread. We are making all citizens wear masks and social distancing. Stopping all international flights"
  - levelTwoStrategy- " is spreading faster. We are forced to close public spaces."
  - levelThreeStrategy - " is getting a bit more dangerous. Stopping all domestic flights. Banning the sale of alcohol and tobacco(shutting down the strip)"
  - levelFourStrategy -" is dangerous. Enforcing a strict lockdown."
  - levelFiveStrategy -" is extremely dangerous. Deploying the army. Shutting down the economy"

3.2 Updating the Pandemic Class. The Pandemic class should be updated with the following: (10)

- Private
  - add the appropriate private member variable to the Pandemic class so it can fulfil the role of the Context participant in the Strategy design pattern.
- Public
  - takeAction- a method that takes no parameters and does not have a return value. This method utilises the Pandemic's specific PandemicStrategy to output a string consisting of the pandemic's name, along with the PandemicStrategy's unique sentence, as described in the previous part.

- increaseLevel - This method should be modified to also change the strategy as the state changes
- decreaseLevel - This method should be modified to also change the strategy as the state changes
- Add a function to allow the setting of a Pandemic initial state and strategy

Task 4: Section A Main .... (15 marks)

4.1 Create a main function that can create a pandemic with a specific name, state and strategy, and then demonstrate your implementation of the State and Strategy design patterns by using the pandemic's functions to show how a pandemic's state and strategy can be changed. You are required to provide a UML Class diagram for the entire Section B and a State diagram that corresponds to your main. All design patterns must be clearly marked in the UML Class diagram. (15)

Section B

In this section you will be creating pixel art for posters to be placed in public spaces which requires you to use the Decorator and Composite design patterns. The Composite design pattern is used to create basic posters as signs to indicate that the public should wear their masks and other PPE in public spaces. The Leaf participants of the Composite are an array of individual elements. The Composite itself can be represented as a row of pointers to components, which could be a composite or a Leaf. The Decorator design pattern is then used to decorate those basic posters.

Task 5: The Image Hierarchy ....(10 marks)

For this task you would need to create the following classes:

5.1 1. **Image**: This class would be the component class for both the Decorator and the Composite design pattern. (10)

2. **ImageElement**: This will have an array of chars which would hold pixels for a row in an image. You may choose the width of the image.

3. **BaseImage**: This will be the Composite class, which will have an array of Image pointers. This array will determine the height of the image.

   (a) Note: Each element of the array should point to an ImageElement.

4. **ImageAdditions**: This will be your Decorator class, which will have the following subclasses.

   (a) Mask

   (b) TemperatureCheck

   (c) UseHandSanitiser

   (d) WashHands

Task 6: Decorator and Composite ....(20 marks)

In this task, you will be implementing the classes created in Task 5.

6.1 1. You will be required to **write your own main and makefile** for this section. (20)

2. The base function should be called drawPoster which will display an empty poster in the terminal. Your poster would need an appropriate border to indicate the poster space in the terminal. You may use special characters of your choosing,examples are: $*$, /, -, +, └, ┌, |

3. Each concreteDecorator class should add an element to the image with the drawElement function.

4. You may use the following links to learn how to colour your posters:

   (a) https://stackoverflow.com/questions/2616906/how-do-i-output-coloured-text-to-a-linux-terminal

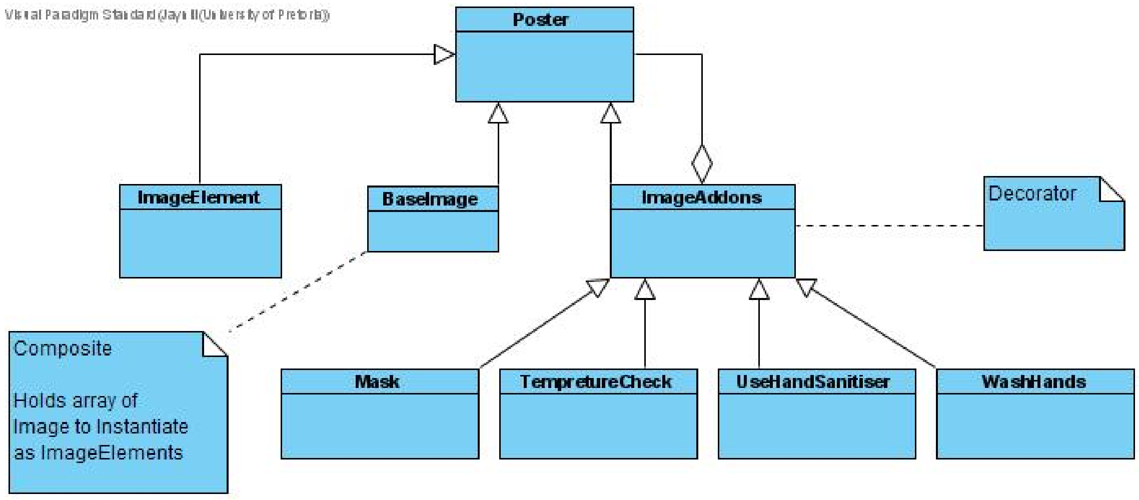
   (b) https://github.com/ikalnytskyi/termcolor

   (c) https://www.lihaoyi.com/post/BuildyourownCommandLinewithANSIescapecodes.html

   (d) There are many useful links on the internet on how to create styling on the terminal should you wish to be creative.

5. You may create the poster however you wish, as long as the elements of the decorator are clearly identifiable.

6. Here is a partial class diagram of the structure of the system.


Visual Paradigm Standard (Jay n ll(University of Pretoria))
Poster
ImageElement
BaseImage
ImageAddons
Decorator
Composite
Holds array of Image to Instantiate as ImageElements
Mask
TempretureCheck
UseHandSanitiser
WashHands

Section C

As a final challenge, you will be required to link Section A and Section B together. Where each level from Section A will influence how the poster in Section B would look. The following needs to be implemented:

Task 7: Updating the Pandemic Class .... (10 marks)

Each level in the pandemic should have it's own poster corresponding to the level that is currently in place. As the increaseLevel and decreseLevel methods are called from Section A, those methods should call the relevant methods to create the poster. You may create additional helper classes and patterns as you feel fit for this section.

7.1 You are required to create a Poster variable in the Pandemic class which will be used to call the necessary functions to create your poster according to the current state of the pandemic. (5)

7.2 When increaseLevel or decreaseLevel is called the poster should be updated to the new poster and printed. (5)

Task 8: Main .... (15 marks)

8.1 This task requires you to update what you have done in Task 4 to indicate the implementation that you have done in this section. You should update your main, State diagram and UML Class diagram. All design patterns must be clearly marked in the UML Class diagram. (15)