Final Part 5 of 6

Started: Aug 19 at 5:24pm

Quiz Instructions

Welcome to the CSC207 Summer 2021 final assessment. This final assessment comes in 6 parts. Each one is a different quiz on Quercus. You have until 14:00 EST on Friday 20 August 2021 to submit your answers to all of them.

SAVE YOUR ANSWERS IN A TEXT FILE ON YOUR COMPUTER before copying them into Quercus and clicking submit.

You can use spell-check, grammar-check, IntelliJ, the internet, and your notes from the course to answer these questions. IF YOU COPY SOMEONE ELSE'S WORDS it is considered to be CHEATING!!!! **Be sure to put everything in your own words.** Give enough detail so that we are convinced that you understand the concepts.

To ask a question during the exam, go to our usual lecture Zoom session (**link (https://utoronto.zoom.us/j/89521849618)**) during the following times:

- 14:00--16:00 ET on Thursday 19 Aug
- 22:00--23:59 ET on Thursday 19 Aug
- 9:00--11:00 ET on Friday 20 Aug
- 13:00--14:00 ET on Friday 20 Aug

The first question in Final Part 1 of 6 contains the statement of academic integrity - please make sure to read it.

Question 1 5 pts

The goal of this question is for you to show us that you understand how your chosen design pattern works and how to implement it. You can discuss Phase 1 or Phase 2 in your answer, but not both.

Choose a design pattern that your team implemented (or that you wanted to implement) in your project, this semester, from this list: Observer, Strategy, Factory Method, Abstract Factory, Builder, Façade.

- Which design pattern did you choose?
- Explain how the design pattern works in general.
- Explain how the design pattern was implemented in your project:

- Which classes were involved?
- What were their names and responsibilities?

If your team did not implement any design patterns, you can instead choose a design pattern and explain why it would have been a bad idea to implement it, given your particular design.

- Which design pattern did you choose?
- Explain how it works.
- Explain why that would not have been appropriate for your project's design.

Edit View Insert Format Tools Table

12pt Paragraph

I used Façade design pattern in the Controller layer named Façade. In general, each responsibility of the Façade class is encapsulated inside a class variable. Every method is in the original class.

In my project, the Façade involves

All the Controller classes: LoginSystem (for user's sign up and login), MessagePublisher (manages the observers which are the listeners), MessageSender (contains the constructor of MessageSender with an initialized event), ScheduleSystem (a controller of ScheduleManager), SendMsgListener (an observer class which implements MessageListener interface), TemplateSystem(a controller of

p 176 words

Question 2

1 pts

For this question, select all answers that apply.

Consider a program that needs a class that validates passwords. You have a `PasswordValidator` class from a previous project that you want to reuse, without rewriting any of the code. To make it interact properly with the rest of your program, the Adapter design patterns says to:

- [✓] Create a superclass for `PasswordValidator` so that it can inherit whatever methods it is missing.
- [✓] Create an `AdaptedPasswordValidator` class that contains a variable of type `PasswordValidator`.
- [✓] Create a `PasswordValidatorAdapter` class that extends `PasswordValidator` and overrides methods where necessary.
- [ ] Create an IPassword interface that is implemented by `PasswordValidator`.

Question 3

1 pts

Consider the following code:

```
public class GameBoard {

  private Player player1;
  private Player player2;
  private WinValidator wv;

  public GameBoard(Player p1, Player p2, WinValidator wv) {
    player1 = p1;
    player2 = p2;
    this.wv = wv;
  }

  public void player1turn(int n) {
    p1.takeTurn(n);
    wv.validateBoard(this);
  }

  public void player2turn(int n) {
    p2.takeTurn(n);
    wv.validateBoard(this);
  }
}
```

This code contains a simplified version of the anti-pattern for Observer. If we wanted to implement the Observer design pattern, how would we do it?

The name of the Observer class would be [Player].

The name of the class being observed would be GameBoard .

Question 4

1 pts

A class called `UserSystem` has too many responsibilities. It is replaced with a class that contains variables, each with one of the `UserSystem`'s previous responsibilities. Which design pattern is being implemented here?

- ○ Strategy
- ○ Adapter
- [✓ Façade]
- ○ Observer
- ○ Dependency Injection

Question 5

0 pts

Explain your answer to the previous question.

Edit View Insert Format Tools Table

12pt Paragraph

In this case, we are supposed to have one class that contains so many responsibilities. each responsibility of the Façade class is encapsulated inside a class variable and every method is in the original class. That's the reason why we choose Façade.

p ▸ span 42 words </>

Quiz saved at 6:51pm Submit Quiz