Homework 3

Similarities and Differences of the Patterns

CSE 4361-001

Dr. David C Kung

Andrew Duong

November 12, 2019

1a. Similarities of the Command and Object Adapter Pattern

Command pattern and object adapter pattern have similarities in how to they both keeps track of patterns and objects that have similarities. Command pattern place the order of similar objects by stacking them on top of each other or in a queue. Object adapter pattern bases the object of a common object by re-using a template or adapting the template. Both Command and object adapter patterns have similarities in where they both rely on one pattern or template to carry on the objective. The command pattern relies on one pattern or template to allow queuing, scheduling, stacking, undoing, or redoing for keeping track of multiple objects of the same exact pattern whereas the object adapter pattern also relies on one pattern or template to allow multiple objects of similar layout to based off one pattern or template that would use the same exact attributes instead of having multiple objects with similar attributes. Both patterns are similar because they use the same pattern or a template.

1b. Differences of the Command and Object Adapter Pattern

Command pattern and object adapter pattern also have differences in how they both keep track of patterns and objects that have similarities. Command pattern place the order of similar objects by stacking them on top of each other or in a queue. Object adapter pattern bases the object of a common object by re-using a template or adapting the template. The differences between the command pattern and the object adapter pattern is that they manage similar objects or templates differently. The command pattern relies on one pattern or template to allow queuing, scheduling, stacking, undoing, or redoing for keeping track of multiple objects of the same exact pattern meanwhile the object adapter pattern relies on one pattern or template to allow multiple objects of similar layout to based off one pattern or template that would use the same exact attributes instead of having multiple objects with similar attributes. Both patterns a different because they solve different problems as one is for queuing, stacking, or scheduling while the other one is just adapting to the template.

1c. Example of Command Pattern Used in a Situation (Assembling and Disassembling Laptops)

When assembling and disassembling laptops, it is important to keep track which parts have to be put together first before the next parts are put together. Like the motherboard and cooling unit has to be assembled first before the hard drive, RAM, and battery components are assembled. Then the hard drive, RAM, and battery components would have to be disassembled first before the motherboard and cooling unit is disassembled. That’s where command pattern is very important in assembling and disassembling laptops. The adapter pattern cannot be applied when the command pattern is applied here. It doesn’t make sense to adapt anything similar here when you’re stacking components.

1d. Example of Object Adapter Pattern Used in a Situation (Managing Models of Laptops)

There are many different types of models of laptops produced and released each year or two by Dell, HP, Lenovo, and other PC manufacturers. There are ways laptop models can be tracked and classified as classes and/or series instead. For Dell, they have different model laptops such as the Inspiron, Latitude, XPS, Precision, Vostro, and more. Each model has series such as 3000, 5000, and 7000 series. Almost each of these models have 3000, 5000, and 7000 series. Depending on the model and series, each model is also based-off the common screen sizes, 11.6-inch, 12.5-inch, 13.3-inch, 14.0-inch, 15.6-inch, and 17.3-inch. Therefore, a Dell laptop like the Dell Precision 7530 I’m using now isn’t just any model of a Dell laptop in general. There’s a pattern or template of a series of a model it’s based-off. First, it’s a 2018 model, so it’s a Dell Precision 7530. There’s also 2017 model which is a Dell Precision 7520. The 2015 model is a Precision 7510 (also called Precision 15 7000). The 2019 model or latest model is a Precision 7540. The 2020 model can easily be predicted as a 7550. See the how the pattern is adopted in the model numbers. Now that’s the 15.6-inch. There’s also 17.3-inch Dell Precision that’s similar to my laptop called a Dell Precision 7730 which is also a 2018 model. The 2017 model would be a Precision 7720 and so on. As you can see several patterns are adapted here as Dell creates individual models. They didn’t just give some random model numbers to the Dell Precision 7000 series laptop each year or two for each screen-size model they created. The 2nd digit in 7x00 is used to classify the screen size. ‘5’ for the 15.6-inch and ‘7’ for the 17.3-inch. The 3rd digit is used to classify the year it’s released. ‘1’ for 2015, incremented to ‘2’ for 2017, and so on. Dell will eventually roll the numbers over or add an extra digit or add a letter to the model name when the reach the last number of the digit in order to create a new template for successor to the Dell Precision 7000 series laptops. The Dell Precision 7000 series is the successor to the Dell Precision M series laptops. The Precision M series is a series to the Dell Precision models released before the Precision 7000 series were introduced and they also used similar strategy of the object adapter pattern when they chose their model numbers for the 15.6-inch and 17.3-inch they released each year or two. It doesn’t make any sense to

2a. Similarities Between Flyweight, Singleton, and Prototype

A flyweight pattern solves several instances of an object by reusing the structure of an object that’s immutable. A singleton pattern solves one single instance of an object by reusing it once that’s mutable. A prototype solves several instances of an object by reusing the structure of an object that’s mutable.

The similarities between Flyweight and Prototype is that it solves several instances of an object by reusing the structure of an object, whether it’s mutable or not. The similarities between Singleton and Prototype is that solve an instance of an object that’s mutable.

2b. Differences Between Flyweight, Singleton, and Prototype

A flyweight pattern solves several instances of an object by reusing the structure of an object that’s immutable. A singleton pattern solves one single instance of an object by reusing it once that’s mutable. A prototype solves several instances of an object by reusing the structure of an object that’s mutable.

The differences between Flyweight and Prototype is that Flyweight solves several instances of an object that’s immutable while Prototype solves several instances of an object that’s mutable. The differences between Singleton and Prototype is that Singleton solves only one single instance of an object that’s mutable while Prototype solves several instances of an object that’s mutable.

2c. Example of a Flyweight

Taking a system image of a hard drive on a PC which can be used for system recovery on multiple systems with the same exact hardware only.

2d. Example of Singleton

Taking a system image of a hard drive on a PC which can be used for system recovery on that particular system only.

2e. Example of Prototype

Taking a system image of a hard drive on a PC which can be used for system recovery on many other systems with similar hardware.