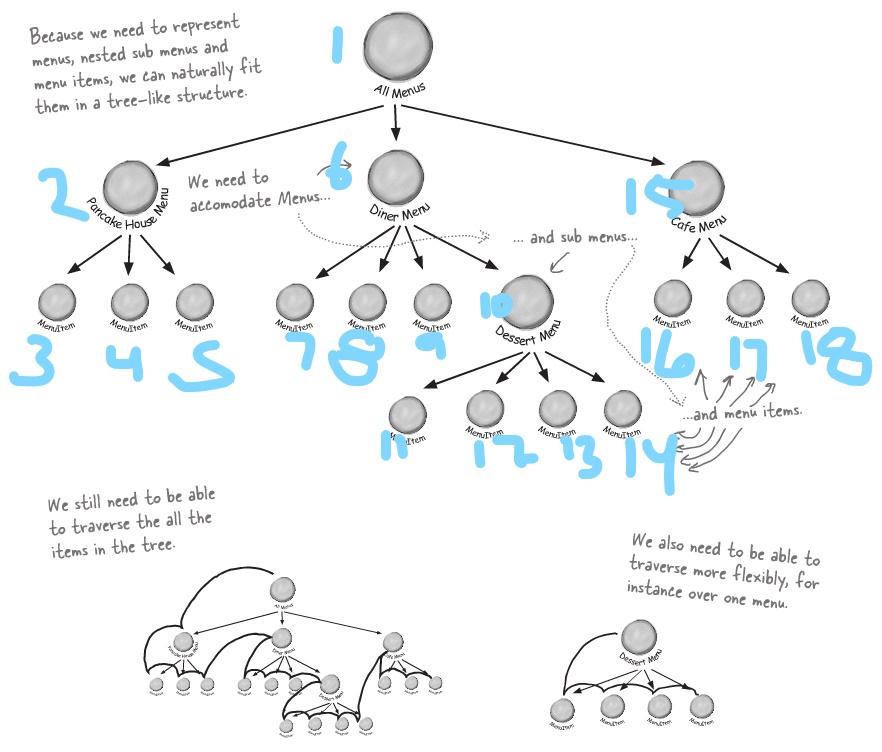
Workout 2019-10-25 –Composite Pattern Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Instructions: Work individually on this workout. Place your answers for questions 1-11 in this word document. Question 12 requires a little bit of programming. Copy the word document containing answers for 1-11 in the root directory of your code. Zip up the code and submit it to ASULearn.

1. How does adding Dessert Menu to the Diner Menu in the book’s example fundamentally change our problem?

-It changes the problem fundamentally by creating a sub menu. It’s no longer linear, we have an item that is in of its self a list of items.

1. Number the following tree in a pre-order ordering. (Number the first node visited as 1, the second as 2, etc.) You can create text boxes in Word, place numbers in those text boxes, and place them over the corresponding tree node.



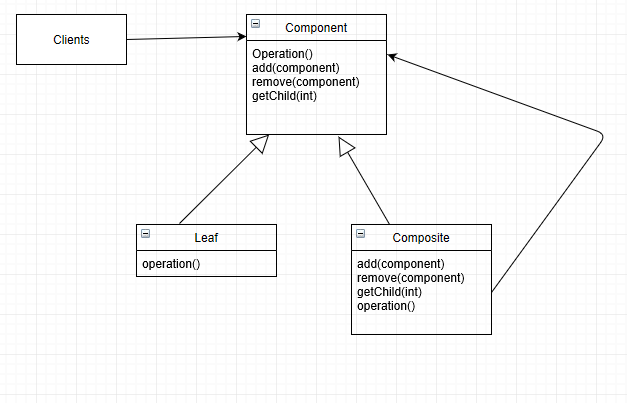
1. State the Composite Pattern.

-Allows you to compose objects into tree structures to represent part-whole hierarchies.

1. Does the Composite Pattern also hide the underlying composite representation from the client (like the Iterator Pattern)? Explain.

-Using a composite structure allows us to apply the same operations over both composite and individual objects which means in most cases we can ignore the differences between composition.

1. Draw the generic UML for the Composite Pattern.



1. How is the Iterator Pattern “embedded” in the Composite Pattern?

The composite pattern uses a tree structure and has the same concept as the iterator when doing an in order traversal.

1. State, as a general rule, what methods should go in the Component abstract class that are used in the Composite Pattern.

- All the methods that are in the composite and leaf.

1. What methods from the Component class are meaningless in the Leaf class? What should happen if the client calls one of these methods on a Leaf object?

-getChild() method and should throw and UnsupportedOperationException.

1. Define transparency? How does it violate the Single Responsibility Principle?

-Transparency allows the client to to treat all classes uniformly, so whether an element is composite or a leaf node it becomes transparent to the client. It breaks the single responsibility principle by allowing a class to take on more than one responsibility.

1. A file on ASULearn contains code that simulates a directory structure in Linux along with a recursive "ls" operator. This version of the Composite pattern sacrifices “transparency” for “safety”. Explain that statement.

It doesn't treat all classes uniformly. In order to access any file or directory you have to go through the root and on down the node leaf hierarchy of the structure.

1. Examine the code and explain how the indention works. Does this approach to indention violate any of our design principles?

-It has a tree node structure so when creating each individual object you can attach it in whatever hierarchical order you want.

1. Copy the code into eclipse (or your favorite IDE). In main, write code that creates the following directory structure (directories below start with uppercase letters, individual files start with lowercase letters). Then run the ls operation on that structure.

Root

Contacts <empty directory>

Documents

fileA.txt

fileB.txt

fileC.txt

Pictures

fileC.jpg

Music

InConstantSorrow.mp3

fileD.txt