Composite Pattern

1. Consider a program with a class for various types of files. For example, there are concrete classes called `TxtFile`, `CppFile`, `PdfFile`,.... Each of those classes has a data member holding a path. There is also a class called `Directory` which also has a data member for a path.

   (a) (5pts) Draw a class diagram showing the proper relationship between files and directories. You should include as many abstract classes as you think is appropriate. Keep in mind that a directory can hold files or subdirectories. Leave space to add a couple methods to each of your classes.

   (b) (5pts) Add a method called `open()` to your file and directory classes. Calling `open()` on a file object should open the corresponding file in the proper application; calling `open()` on a directory should open all the files inside of the directory, along with all the files in any subdirectory. decorate your class diagram with pseudocode for Directory::open().

   (c) (5pts) Now, suppose you want to add a method Directory::search(file) that returns true if and only if the file is inside of the directory or in any subdirectory. Update your class diagram and include all the pseudocode necessary to implement the search method.

2. (a) (4pts) Draw the generic class diagram for the Composite pattern without including child management.

(b) (4pts) Explain how a directed graph[1] is related to the composite pattern.

(c) (6pts) In a Composite pattern, there are often *child management methods* such as `add(child)`, `remove(child)`, `getChild(int)`,.... Some of these methods may not make sense depending on the particular program. There is a choice of whether to include (abstract) child management methods in the `Component` class. Describe the pros and cons related to this decision. Include any relevant design principles in your answer.

[1] A directed graph is (informally) just a picture of a bunch of nodes (dots) and node-to-node arrows.

3. When I look at the home screen of my phone, I see a bunch of application icons. I have multiple pages of these icons. Moreover, on some pages I have an icon representing a collection of applications (I think they are called application folders). When I drag one icon to another page or folder, the icon is removed from it's current location and added to the new one. Also, when I drag an application icon onto another application icon, both app icons are moved to a new folder in the location of the non-dragged icon.

   To model this story, we can have classes with names like `AppIcon`, `AppFolderIcon`,.... We should also have child management methods `add(icon)`, `remove(icon)`, `getParent()`,.... In particular, an object's method `add(icon)` will be called whenever the icon is dragged onto the object.

   (a) (6pts) Draw a class diagram for this situation.

   (b) (4pts) Write down detailed pseudocode for the implementation of `AppIcon::add(icon)`.

4. For this exercise you will need to look in CoCalc at the folder `Composite/List2html`.

   (a) (4pts) Draw the class diagram for the program. Leave plenty of space for adding more classes to the diagram.

   (b) (3pts) Identify the Composite pattern roles for each class in that program:

   `Component`:

   `Composite`:

   `Leaf`:

   (c) (10pts) Calling the method `List::print(outFile)` on a List object is supposed to use the stream `outFile` to print the corresponding unordered list into an html document. For example, with the current setup of `Main.cpp` the program is supposed to create a file `output.html` that looks something like `solution.html`. To get the program to work you need to implement the methods `List::print()` and `ListItem::print()`.

   If you have any questions about streams or html lists, ask!

   (d) (5pts) What design pattern could you use if you want to extend this program so that it can also print lists nicely to a pdf or txt document? Add the appropriate modifications to the class diagram above for this extension. You don't need to implement any extra code.