Reflection Document

1. What did you like about this assignment?
I found that this assignment should be used when clients need to ignore the difference between compositions of objects and individual objects. If anyone is using multiple objects in the same way, and often have nearly identical code to handle each of them, then composite is a good choice. It becomes less complex and helps you handle the composite objects and individual objects homogeneously. It's one of the most significant advantage is that less number of objects reduces the memory usage, and it manages to keep us away from errors related to memory. An added advantage to this is that in java sharing objects is easy which therefore help reduce execution time.

2. Discuss if you have chosen type-safety or transparency in implementing the child management operations of adding a resource to a cluster.
It would be more accurate to say that transparency is valued more over type-safety in implementing operations of adding a resource to a cluster.
Incase of type-safety, declaring methods applicable only to Composite Class than leaf because Component implementers will need to choose individually for leaf or composite class that which methods are supposed to be implemented in leaf and which in Composite. Also, this pattern is chosen because the properties and functionalities of all the classes behave same.
Whereas incase of Transparency, leafs are almost rare. For example, in a UI toolkit, it is easier to add children to any component which normally has little use for children, like a checkbox and expect them to be rendered intelligently. Therefore, rather than making decisions on what to change, it not only gives user but also the programmer an ease to use the code uniformly. Hence, we use Transparency over Type-safety.

3. What did you find difficult?
In my point of view it was pretty easy to implement the Composite Pattern. But, later on it might become extremely difficult to refactor the code for this particular pattern. As in this cutting out methods or refactoring some functionalities in Composite class might be difficult as it would influence the leafs too.

4. Anything that I learned from the assignment I did not know before.
Yes, I learned that how exactly composite objects can bring uniformity in the functionality of leaf and the composite class. Also, that Composite Design

Pattern makes it harder to restrict the type of components of a composite. Therefore, it must not be used incase of a full or partial hierarchy of objects. The greatest takeaway from this assignment was that this pattern makes design too general. Also, sometimes you want a composite to have only certain components but because it involves interface and all the classes must implement it makes it difficult to keep the design type-safe when you want it.