**Composite Design Pattern:**

Advantages:

1. It defines class hierarchies that contain primitive and complex objects.
2. It makes easier to you to add new kinds of components.
3. It provides flexibility of structure with manageable class or interfaces.
4. Lean, east-to-understand program code.
5. Great expandability.

Disadvantages:

1. Implementation of component interfaces is very challenging.
2. Subsequent adjustment of composite features is difficult and cumbersome to realize.
3. Once tree structure is defined, the composite design makes the tree overly general.
4. In specific cases, it is difficult to restrict the components of the tree to only particular types.

**Iterator Design Pattern:**

Advantages:

1. It supports variations in traversal of a collection.
2. It simplifies the interface to the collection.
3. It can achieve dynamic polymorphism.
4. Reduce code duplication
5. Minimizes business logic complexity.
6. Adheres to Open/Closed Principle, you can add new ways to traverse collections without breaking your existing collections or client code.

Disadvantages:

1. You can’t change a collection while traversing it with an iterator.
2. Iterator is overkill for small application.