Exercises on Design Patterns II

1.

(a) Strategy Design Pattern – according to GOF, the Strategy Design Pattern defines a family of algorithms, encapsulating each one, and makes them interchangeable. It lets the algorithm vary independently from clients that use it. It is based on following principles:

- Objects have responsibilities
- Different, specific implementations of these responsibilities are manifested through the use of polymorphism
- There is a need to manage several different implementations of what is, conceptually, the same algorithm.

(b) Strategy design pattern should be used when the selection of an algorithm that needs to be applied depends on the client making the request or the data being acted on. Selection of algorithm is made based upon context.

3. Abstract Design Pattern is appropriate for situations when it is necessary to create families of related or dependent objects without specifying their concrete class. The AbstractFactory defines the interface for how to create each member of the family of objects required. Each family is created by having its own unique ConcreteFactory.

5. Keeping instantiation and initialisation of components of object within object is a suitable strategy for simple objects and for situation when the object construction process is definite and always produces the same representation of the object. This design may not be effective when the object being created is complex and the series of steps constituting the object creation process can be implemented in different ways producing different representation of the object. Because different implementations of the construction process are kept within the object, the object can become bulky (construction bloat) and less modular. Also, adding new implementation or making changes to an existing implementation requires making changes to the existing code. The Builder pattern shifts construction of complex object out of the object class and into a separate builder class. Each builder class can implement different representation of the object. This reduces object size.

7.

(a) The Façade Pattern is a simplified interface that performs many other actions behind the scenes. Sometimes series of steps need to be performed to undertake a particular task. The Façade pattern involves the creation of a separate object that simplifies the execution of such steps.

(b) It should be used when existing interfaces are too complicated or verbose. Facade Pattern aims to make system easier to use.

9. Use Bridge Design pattern when want to vary the implementation and the abstraction by placing the two in separate class hierarchies. It is useful in graphics and windowing system that need to run over multiple platforms.

11.

(a) Composite Design pattern is Structural design pattern, which composes objects into tree structures to represent part-whole hierarchies.

(b) Use Composite Design pattern when need to treat individual objects (leafs) and compositions of individual objects (trees) uniformly. In other words, independent of the type of component, the interface with client should be the same

(c)

- Leaf - has no children, simplest component, individual object.
- Composite - is composition of components, consist of leafs and/or composites.
- Component - defines an interface for all objects in the composition, both the composite and leaf nodes. It may implement default behaviour for the interfaces. Components come in two flavours: composites and leaf elements.
- Client – user of the Component interface.