Exercises on Design Patterns II

Patterns: ***Strategy***, ***Abstract Factory***, ***Builder***, ***Facade***, ***Bridge***, ***Composite***.

- Please see src file in my repo for the coding questions

1.

(a) Briefly describe the Strategy Design Pattern

- The strategy pattern defines a set behaviours (or algorithms) that can be used interchangeably. Strategy lets the algorithm vary independently from the clients that use it.

(b) When is it appropriate to use the Strategy Design Pattern?

- When you have a class or interface that may be extended/implemented in various ways.
- When you want your class to be open for extension but closed for modification

3. When is it appropriate to use the Abstract Factory design pattern?

- When you wish to create families of related objects, without specifying a concrete class. The abstract factory pattern is like a 'factory of factories'.

5. "In general, the details of object construction, such as instantiating and initialising the components that comprise the object, are kept within the object, often as part of its constructor."

Comment on this statement with reference to *modularity* and *construction bloat*.
Keeping details of object construction within the object may not be very efficient when a complex object is created and its creation can be implemented in different ways that produce different representations. As different representations of the construction process are kept within the object it can become bulky, which is known *as construction bloat*, and also less *modular*.

7.

(a) What is the Facade Pattern?
The façade pattern provides a higher-level interface to a subsystem of classes, making the subsystem easier to use.

(b) When, and why, would you use the Facade Pattern?
You would use the façade pattern when you have a system that is very complex and difficult to understand. You would use it when because it hides the complexities of the larger system and provides a simpler interface to the client.

9. When should one make use of the Bridge Design Pattern?
The bridge pattern should be used when you wish to decouple an abstraction from its implementation so that the two can vary independently.

11.

(a) What is the Composite Pattern?

The composite pattern composes objects into tree structures to represent part-whole hierarchies. Implementing this pattern allows clients to treat individual objects and compositions uniformly.

(b) Under what conditions would you use a Composite Design Pattern?

When you have a group of objects that you want to be treated in the same way asa a single instance of an object

(c) What are the four participants of the Composite Design Pattern?

1. Component – The abstraction of leafs and composites
2. Leaf – Objects that have no children
3. Composite – Stores child components
4. Client – Manipulates objects using Component interface