23311 Software Engineering 1

BEHAVIORAL	**SRTUCTURAL**	**CREATIONAL**
Chain of Responsibility	***Adaptor***	Abstract Factory
Command	Bridge	Builder
Interpreter	***Composite***	***Factory Method***
Iterator	Decorator	Prototype
Mediator	Façade	***Singleton***
Memento	Flyweight	
Observer	Proxy	
State		
Strategy		
Template Method		
Visitor		

Strategy Pattern (Behavioural) provide a set of objects that hold alternative algorithms to solve a problem. Strategy could not be changed at runtime. New strategy could be easily added. Common behaviours may be defined in the interface.

State Pattern (Behavioural) provide a set of objects that hold algorithms for different states. Adding a new state requires adding a new class. State-specific behaviour defined in each class.

Composite Pattern (Structural) provide a uniform base and then implement the base through multiple classes. Should be considered when data or code resembles to a tree. It has three levels – ***component*** (element), ***composite*** (folder), and ***leaf*** (file)

Adapter (Structural) provide a class that utilises another class to provide a richer functionality compared to the original class (***adoptee***).

Factory Method (Creational) provides a way for creating objects that has the same base and is useful when a lot of classes share a same base. It separates the object creation (by this method) from the creation interface (pre-defined and as parameters).

Singleton Pattern (Creational) provides a globally unique and accessible class. It is possible to do an antipattern and make it not suit these properties but having its functions and contents as-is.

Immediately after making a change to a code, it is needed to search throughout the code base for similar codes and change accordingly.