**Behavioural:**

**Strategy Pattern:**

The Strategy Pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

Take the parts that vary and encapsulate them, so that later you can alter or extend the parts that vary without affecting those that don’t.

**Observser Pattern:**

The Observer Pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

**Decorator Pattern:**

**OO Design Principles:**

* Favour composition over inheritance
* Encapsulate what varies.
* Program to interfaces, not implementations
* HAS-A is better than IS-A
* Strive for loosely coupled designs between objects that interact.

**Notes:**

* Built in classes are available for observer pattern.
* Static methods can’t be overrided.