**Aggregate Root** Design Pattern: Objects are treated as single unit and they go through a parent / root object. For every operation of that class has to be controlled in the same class only. It is used for order-order items, customer-address, addresses-phones.

**Iterator Pattern**: Provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation. Accesses elements in a safe way, that is it will not allow to manipulate but it can only loop

**Iterator Pattern** connects with **Aggregate Root**. We can use enum for collection instead of list in aggregate root.

**Adapter Design pattern**: It makes two incompatible classes work together. When we have two classes whose interfaces are not same and we want to make them compatible, we put an adapter in between and then we make a call.

**Template Pattern**: Define the skeleton of an algorithm in an operation, deferring some steps to client subclasses.

**Bridge Pattern:**