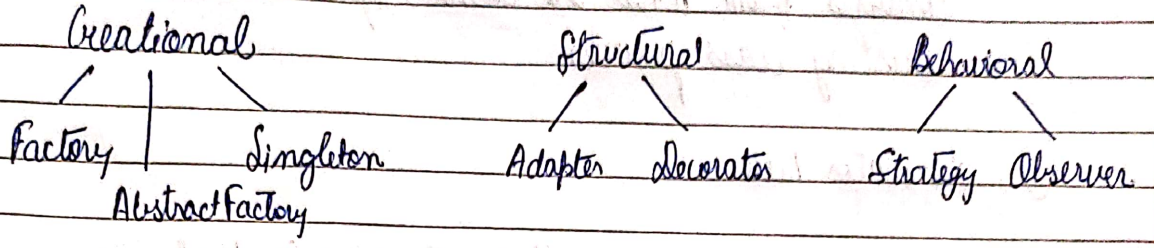
Lecture #4

Creational Patterns - How object are created
Structural Patterns - How class/object are composed
Behavioral Pattern - How object behave



① Factory Pattern -

Suppose we want to have an sms/email notification sender. We can move it inside a factory.

```
auto emailNotif = NotificationFactory::create("email");
emailNotif -> send("welcome", "sarthak@gmail.com");
```

② Abstract Factory -

Factory for one service like notification
Abstract factory for more than 1 service like createButton, createCheckBox (set of related products)

③ Singleton -

Ensure exactly one instance of a class exists, and we have global access to it.
Downsides - ① Anyone anywhere in the code can access it & modify its state
② ~~Tight coupling~~

Structural Pattern

① Adapter Pattern -

Adapter lets you use an existing class work with a different interface by wrapping it. We don't touch the old class, we write the wrapper that converts from what the client expects → what the existing class provides.

② Decorator Pattern -

You wrap an object with another object that implements the same interface and adds behaviour before/after delegating to the wrapped object.

Behavioral Patterns

① Strategy Pattern

Encapsulates interchangeable algorithm behind a common interface. Similar to Factory in creational pattern like we have interface PaymentStrategy and CardPayment and UPIPayment implement it. Now we have a class ShoppingCart where we give the strategy and the amount to initiate the payment.

② Observer Pattern

One to many dependency, when subject changes, all observers get notified.
A news ~~channel~~ agency behaves as subject, having a list of observers which are news channels.
We can add new news channels as observers.
Then we can notify all observers with some message.

Scanned with OKEN Scanner