Timeline

Description automatically generated

Creational patterns :

Provides object creation mechanism in a controlled manner that helps reduce code complexity.

Structurla Patterns :

It explains the how to combine the objects and classes into bigger strucrtures while maintaining flexibility and efficiency.

Behavoiural Pattern:

Behavioural Pattern is all about the communication between objects

1. [Adapter Pattern](https://javarevisited.blogspot.com/2016/08/adapter-design-pattern-in-java-example.html)— Acts as a connector between two incompatible interfaces.
2. Bridge Pattern — Decouple an abstraction from its implementation.
3. [Composite Pattern](https://javarevisited.blogspot.com/2018/02/composite-design-pattern-in-java-real.html) — Treat a group of objects similar to a single object.
4. [Decorator Pattern](https://javarevisited.blogspot.com/2011/11/decorator-design-pattern-java-example.html) — This allows a user to add new functionality to an existing object without altering its structure
5. [Facade Pattern](https://javarevisited.blogspot.com/2015/01/adapter-vs-decorator-vs-facade-vs-proxy-pattern-java.html) — Hides the complexities of the system and provides a simpler interface to the client.
6. Flyweight Pattern — Used to reduce the number of objects created and decrease memory footprint and increase performance
7. proxy Pattern — A class represents the functionality of another class