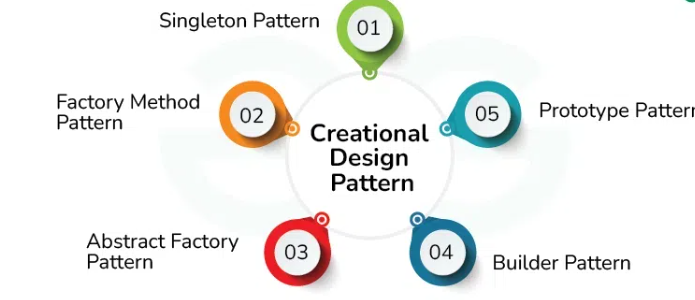
**Design Patterns**

* Design Patterns are reusable solutions to software design problems. They can be used for creating scalable, manageable and scalable software. These can be used as only a blue print or templates only.
* **Characteristics**: Reusability, Standardization, Efficiency and Flexibility.
* **Why Learn them?** They offer solutions that have been tried and tested. Learning them enables to quickly and effectively address similar challenges in various projects. Reduces redundancy and saves development time.
* **Types:**
  + Creational
  + Structural
  + Behavioural
* **Creational**
  + These patterns talk about creation of objects or problems related to object creation. They can make system independent of how it’s objects are created, composed and represented.

****

1. **Factory 🡪** It Provides an interface that creates a factory method that take the responsibility of creating objects. This pattern separates object creation and object implementation that promotes loose coupling. This offers flexibility, extensibility and maintainability.

**When to use?**

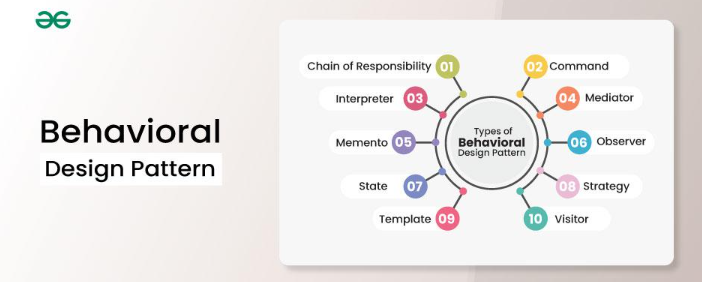
* If object creation is complex or varies on different conditions
* To separate object creation and object implementation that reduces dependencies to make it easier to modify or expand the system without effecting existing code.
* If application needs to create different versions of product this provides flexible way to handle these variations by defining different factory methods for each product type.

**Disadvantages:**

* Adds more classes and interfaces which can complicate maintenance.
* Slight performance impacts due to polymorphism.
* Concrete classes are linked to their products.
* Clients need knowledge of specific subclasses.



* **Structural** 
  + **Bridge 🡪**
* **Behavioural**
* These patterns define how objects interact and communicate with each other. They also define how objects must collaborate and distribute responsibility amongst them, making it easier for complex control flow and communicate with each other.

****

1. **Command 🡪**