Creational patterns involve object instantiation, and all provide a way to decouple a client from the objects it needs to instantiate

|  |  |
| --- | --- |
| Singleton | Ensures one and only object is created |
| Abstract Factory | Allows a client to create families of objects without specifying their concrete classes. |
| Factory Method | Subclasses decide which concrete classes to create |

Behavioural pattern is concerned with how classes and objects interact and distribute responsibility

|  |  |
| --- | --- |
| Template Method | Subclasses decide how to implement steps in an algorithm |
| Iterator | Provides a way to traverse a collection of objects without exposing its implementation |
| Command | Encapsulations a request as an object |
| Iterator | Provides a way to traverse a collection of objects without exposing its implementation |
| Observer | Allows an object to control access to it |
| State | Encapsulates state-based behaviours and uses delegation to switch between behaviours |
| Strategy | Encapsulates interchangeable behaviours and uses delegation to decide which one to use |

Structural patterns let you compose classes or objects into larger structures

|  |  |
| --- | --- |
| Decorator | Wraps an object to provide a new behaviour |
| Composite | Clients treat collections of objects and individual objects uniformly |
| Façade | Wraps an object to control access to it |
| Adapter | Wraps an object and provides a different interface to it |
| Proxy | Simplifies the interfaces of a set of classes |