DESIGN PATTERNS

1. Singleton is a creational design pattern that lets you ensure that a class has only one instance while providing a global access point to this instance.
2. Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.
3. Iterator is a behavioral design pattern that lets you traverse elements of a collection without exposing its underlying representation (list, stack, tree, etc.).
4. Proxy is a structural design pattern that lets you provide a substitute or placeholder for another object. A proxy controls access to the original object, allowing you to perform something either before or after the request gets through to the original object.
5. Dependency Injection design pattern allows us to remove the hard-coded dependencies and make our application loosely coupled, extendable and maintainable. We can implement dependency injection in java to move the dependency resolution from compile-time to runtime.
6. Template Method is a behavioral design pattern that defines the skeleton of an algorithm in the superclass but lets subclasses override specific steps of the algorithm without changing its structure.
7. Model-View-Controller Pattern is used to separate the application's concerns.

* Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update the controller if its data changes.
* View - View represents the visualization of the data that the model contains.
* Controller - Controller acts on both model and view. It controls the data flow into the model object and updates the view whenever data changes. It keeps the view and model separate.

1. Data Access Object Pattern or DAO pattern is used to separate low-level data accessing API or operations from high-level business services. Following are the participants in Data Access Object Pattern.

* Data Access Object Interface - This interface defines the standard operations to be performed on a model object(s).
* Data Access Object concrete class - This class implements the above interface. This class is responsible to get data from a data source which can be a database / XML or any other storage mechanism.
* Model Object or Value Object - This object is a simple POJO containing get/set methods to store data retrieved using DAO class.

1. In the Null Object pattern, a null object replaces the check of the NULL object instance. Instead of putting the check for a null value, the Null Object reflects a do-nothing relationship. Such a Null object can also be used to provide default behavior in case data is not available.

In the Null Object pattern, we create an abstract class specifying various operations to be done, concrete classes extending this class and a null object class providing do-nothing implementation of this class and will be used seamlessly when we need to check the null value.

1. Façade is a structural design pattern that provides a simplified interface to a library, a framework, or any other complex set of classes.
2. Builder is a creational design pattern that lets you construct complex objects step by step. The pattern allows you to produce different types and representations of an object using the same construction code.
3. Factory Method is a creational design pattern that provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects that will be created.
4. State Pattern says that "the class behavior changes based on its state". In State Pattern, we create objects which represent various states and a context object whose behavior varies as its state object changes.

The State Pattern is also known as Objects for States.