**Design Patterns**

**MVC Pattern:**

MVC Pattern stands for Model-View-Controller Pattern. It individually performs actions on application's. Each component concerns about its own task. Altogether 3 components completes the application.

**Model:** It refers to a object and also can consists of the application logic.

**View:** It is used to view the output of the application.

**Controller:** It takes input from user and sends requests to the MODEL/VIEW accordingly. It can control the data flow in Model module and also changes the view in View module correspondingly.

**Advantage of Model**

Navigation control is centralized Now only controller contains the logic to determine the next page.

Easy to maintain

Easy to extend

Easy to test

Better separation of concerns

**Disadvantage of Model**

We need to write the controller code self. If we change the controller code, we need to recompile the class and redeploy the application.

**Implementation:**

**Singleton Pattern:**

It allows to create only a single object in a single class. This class gives way to access its only object which can be accesseddirectly without need to instantiate the object of the class.

There are a few things to remember whenever you create a singleton class:

* The constructor needs to be private, to prevent the possibility of other objects creating instances of your class.
* In Java, build a Singleton using an Enum.
* JEE 7 has a built-in annotation named @Singleton, along with other related annotations.
* In frameworks such as Spring, the objects that are managed are called beans, and beans are singletons by default. What Spring does well is ensure that all of this is in the background.

**Advantage :**

Saves memory because object is not created at each request. Only single instance is reused again and again.

**Disadvantage :**

The main disadvantage of using the singleton pattern is that the resulting code is difficult to unit test. Make a clear decision as to where you absolutely need to use a singleton, and where you don’t.

**Implementation:**

**Factory Method Pattern:**

This pattern create parent object in interface/Abstract class then child classes are responsible to which class to instantiate.

**Advantage:**

* It greatly simplifies the task of object creation.
* Factory Method Pattern allows the sub-classes to choose the type of objects to create.
* It promotes the loose-coupling by eliminating the need to bind application-specific classes into the code. That means the code interacts solely with the resultant interface or abstract class, so that it will work with any classes that implement that interface or that extends that abstract class.

**Implementation:**

**Abstract Factory pattern:**

This pattern create parent factory method object in interface/Abstract class above child factory methods. In other words it is creating super-factory method pattern upon factories method pattern.

**Advantage**

* Abstract Factory Pattern isolates the client code from concrete (implementation) classes.
* It eases the exchanging of object families.
* It promotes consistency among objects.

**Implementation:**