Design Patterns - Difference Between FAQs

1. Difference between Factory Pattern and Abstract Factory Pattern

S.No	Factory Pattern	Abstract Factory Pattern
1	Create object through inheritance	Create object through composition
2	Produce only one product	Produce families of products
3	Implements code in the abstract creator that make use of the concrete type that sub class produces	Concrete factories implements factory method to create product

2.Difference between Abstract Factory Pattern And Builder Pattern

S.No	Builder Pattern	Abstract Factory Pattern
1	In Builder Pattern, there will be one Director class which will instruct Builder class to build the different parts/properties of our object and finally retrieve the object.	Abstract Factory Pattern will return the instance directly.
2	It will have reference to the created object.	It does not keep the track of it's created object.

3.Difference between Builder Pattern And Composite Pattern

S.No	Builder Pattern	Composite Pattern
1	It is used to create group of objects of predefined types.	It creates Parent - Child relations between our objects.

4.Difference between MVC and MVP

S.No	MVP	MVC
1	MVP is a bit more complex to implement than MVC .Also, it has additional layer for view interfaces.	MVC is easier to implement than MVP.
2	The request is always received by the View and delegated to the presenter which in turn gets the data does the processing	The request is received by the controller which in turn gets the required data and loads up the appropriate view
3	The presentation and view logic an be unit tested as the	The controller logic can be unit tested.

	view is loosely coupled.	**Note:** We can unit test view if we are using Razor view engine. ASPX viewengine are not that easily unit testable
4	MVP is best suitable for Windows Programming as the flow naturally tend towards this pattern.	MVC is best suitable for Web Programming.

5.Difference between Proxy Pattern and Observer Pattern

S.No	Proxy Pattern	Observer Pattern
1	The Proxy Pattern is used for wrapping a kind of special object with 1 or more other objects.	The Observer Pattern is used by a publisher object to notify subscriber objects with information.
2	Either because we don't always have the wrapped object or because it needs to be managed in a certain way that can be simplified via the proxy object(s). This is kind of a way to exchange the API of an existing class with a proxy class. We are not just talking events here, but whatever kind of functionality to define via the proxy object instead of the real object.	The publisher object does not know the subscribing objects - except that the conform to a certain subscriber interface. This is a very flexible pattern for distributing events, since those that wants to listen on certain object has the power to do so without changing the code of the publishing object.

6. Difference between Singleton Pattern and static class

S.No	Singleton Pattern	static class
1	Singleton pattern maintains single instance.	We cannot create instance for static class.
2	A singleton can extend classes and implement interfaces.	A static class cannot . **Note:** It can extend classes, but it does not inherit their instance members.
3	A singleton can be initialized lazily or asynchronously.	A static class is generally initialized when it is first loaded, leading to potential class loader issues.
4	Singletons can be handled polymorphically without forcing their users to assume that there is only one	static class cannot be handled polymorphically.

	instance.	
5	Singleton Class can have value when Class object instantiated between server and client, such a way if three client want to have a shared data between them Singleton can be used.Thats why singleton class can be used for state mangement in stateless scenarios like shopping cart scenario.	Static are always just shared and have no instance but multiple references.
6	We can pass singleton object as parameter	We cannot pass parameter in static class
7	Singleton provides flexibility and also provides sort of a mechanism to control object creation based on various requirements. They can be extended as well if need arises. In other words we are not always tied to a particular implementation. With Singleton we have the flexibility to make changes as when situation demands.	Static classes once defined could not accomodate any future design changes as by design static classes are rigid and cannot be extended.

7.Difference between Strategy and Inversion of Control (IOC)

S.No	Strategy Pattern	Inversion of Control (IOC) Pattern
1	The strategy pattern is useful when we want classes to depend on the interface rather than the implementation.And we can easily swap out behavior depending on which concrete implementation we provide.	Inversion of Control/Dependency Injection (IoC/DI) comes into play when we want the concrete strategy implementation injected into a class. For example, we could use the DI Framework Ninject and configure it so that it will know which concrete strategy implementation to inject into a class in specific scenarios. **Note:** Strategy is just one of the ways that IOC is implemented

8.DIfference between IDictionary and Dictionary

S.No	IDictionary	Dictionary
1	IDictionary is just a contract, abstraction	Dictionary is concrete implementation.
2	It is recommended for example to expect as argument an IDictionary rather than concrete Dictionary, or to expose property of IDictionary rather than Dictionary, because this promotes loose coupling. Than we are able to change underlying objects in the future without affecting those who use your object.	Argument or Property is not required for Dictionary.

9.Difference between Factory Pattern and Dependency Injection

S.No	Factory Pattern	Dependency Injection(DI)
1	Factory is used to create objects	DI is used to move the responsibility of creating an object outside the main code.
2		Some of the well known framework available for DI are 1. Unity Application Block (Microsoft) 2. Ninject 3. StructureMap 4. Castle Windsor 5. Munq/Funq 6. Autofac

10.Difference between String.Clone() and String.Copy() method

S.No	String.Clone()	String.Copy()
1	Returns a reference to this instance of String. i.e., it gives pointer value(ie Current memory Reference)	Creates a new instance of String with the same value as a specified String. i.e., it creates an instance in Heap Memory and gives pointer value(ie New Memory Reference)

11.Difference between Strategy Pattern and Factory Pattern

S.No	Strategy Pattern	Factory Pattern
1	Strategy's sole intent to is to provide a mechanism to select different algorithm.	Factory's sole purpose is to create objects .
2	We cannot use "strategy" to create objects of "factory".	We can use "factory" to create objects of "strategy".

12.Difference between Proxy and Adaptor

S.No	Proxy Pattern	Adaptor Pattern
1	Proxy on the other hand represents a standin object for the real object. This is required when the real object is complex to create, is not available, or for authentication purpose. For e.g. web service proxy, proxy authentication server etc. Proxy can be categorized into Virtual Proxy Remote Proxy Protection Proxy	Adapter is used to adapt to incompatible interfaces. It's more directed from the client (those who consume the object) perspective. A client expects an interface to be of particular type and adapter plays a role in filling that gap. It's more from making it easier for the client to adapt to other third party libraries within there own by adapting to it.

13.Difference between Decorator and Visitor

S.No	Decorator Pattern	Visitor Pattern
1	Decorator may have just a single object to customize.	Visitor has a tree of objects to be worked upon.
2	Decorator does not require a traverser for successful implementation.	Visitor requires a traverser for successful implementation.
3	Decorator pattern is a structural pattern that help us to add new function to an object in the run time , note that in the run time not design time .	Visitor pattern is Behavioral pattern that seperate the data structure from the operation (functionality) that work on it , this mean we can add different operation on the same data structure

Reference: http://onlydifferencefaqs.blogspot.in/2012/07/design-patterns-difference-

between-faqs.html

Difference between MEF and DI / IOC

Difference between MEF and DI / IOC

S.No	MEF	DI / IOC
1	**Abbreviation:** MEF stands for Managed Extensibility Framework	**Abbreviation:** DI / IOC stands for Dependency Injection and Inversion Of Control
2	**Meaning:** Managed Extensibility Framework (MEF) is a component of .NET Framework 4.0 for creating lightweight, extensible applications. It allows application developers to discover and use extensions with no configuration required. It also lets extension developers easily encapsulate code and avoid fragile hard dependencies. MEF not only allows extensions to be reused within applications, but across applications as well. MEF was introduced as a part of .NET 4.0 and Silverlight 4.	**Meaning:** Dependency injection is a software design pattern that allows a choice of component to be made at run-time rather than compile time. This can be used, for example, as a simple way to load plugins dynamically or to choose mock objects in test environments vs. real objects in production environments. This software design pattern injects the dependent element (object or value etc) to the destination automatically by knowing the requirement of the destination.Inversion of Control usually refers to the "containers" while Dependency Injection refers to the actual pattern.DI is a form of IoC.Dependency injection is the main method to implement Inversion of Control.
3	**Where it can be used ?** MEF is useful for dynamic dependencies.	**Where it can be used ?** IOC is most useful with Static dependencies
4	**Used for Known / Unknown parts:** MEF is for discovery of unknown parts.	**Used for Known / Unknown parts:** IOC is for registration of known parts.
5	**Best choice for 3rd party controls or not:** When working with/for 3rd party dlls , MEF is more the best choice.	**Best choice for 3rd party controls or not:** When working with/for 3rd party dlls , IOC is not the best choice.
6	**Swapping DLLs at runtime possible or not:** If we have two dlls, taking a	**Swapping DLLs at runtime possible or not:** If we have two dlls, taking a

	decision at runtime to swap the two is possible using MEF.	decision at runtime to swap the two is not possible using IOC.
7	**Plug-in versioning problem:** Plug-in versioning problem is not there in MEF	**Plug-in versioning problem:** Plug-in versioning problem is there in IOC

Reference: http://onlydifferencefaqs.blogspot.in/2012/09/difference-between-mef-and-di-ioc.html

RUP vs SCRUM methodologies

Difference between RUP and SCRUM methodologies

S.No	RUP	SCRUM
1	**Cycle:** Formal Cycle is defined across 4 phases, but some workflows can be concurrent .	**Cycle:** Each sprint (iteration) is a complete cycle.
2	**Planning:** Formal project plan, associated with multiple iterations, is used. The plan is end-date driven and also has intermediate milestones.	**Planning:** No end-to-end project plan. Each next iteration plan is determined at the end of the current iteration (NOT end-date driven). Product Owner (Key Business User) determines when the project is done.
3	**Scope:** Scope is predefined ahead of the project start and documented in the Scope document. Scope can be revised during the project, as requirements are being clarified, but these revisions are subject to a strictly controlled procedure.	**Scope:** Instead of scope, SCRUM uses a Project Backlog, which is re-evaluated at the end of each iteration (sprint).
4	**Artifacts:** Vision/Scope Document, Formal functional requirements package, system architecture document, development plan, test plan, test scripts, etc.	**Artifacts:** The only formal artifact is the operational software.
5	**Type of Project/Product:** Recommended for large, long-term, enterprise-level projects	**Type of Project/Product:** Recommended for quick enhancements and organizations

	with medium-to-high complexity.	that are not dependent on a deadline.

Summary: Both methodologies are considered to be Agile and approach project activities in the iterative way.

Reference: http://onlydifferencefaqs.blogspot.in/2012/09/difference-between-rup-and-scrum.html

Difference Between Flowchart and Data Flow Diagram

Difference Between Flowchart and Data Flow Diagram (DFD)

S.No	Flowchart	Data Flow Diagram (DFD)
1	Flow chart presents steps to complete a process	Data flow diagram presents the flow of data
2	Flow chart does not have any input from or output to external source	Data flow diagram describes the path of data from external source to internal store or vice versa.
3	The timing and sequence of the process is aptly shown by a flow chart	The processing of data is taking place in a particular order or several processes are taking simultaneously is not described by a data flow diagram.
4	Flow chart shows how to make a system function.	Data flow diagrams define the functionality of a system
5	Flow charts are used in designing a process	Data flow diagram are used to describe the path of data that will complete that process.
6	We have following types of flowcharts, • System flow chart • Data flow chart • Document flow chart • Program flow chart	We have following types of data flow diagrams, • Physical data flow diagrams • Logical data flow diagrams

Reference: http://onlydifferencefaqs.blogspot.in/2012/08/difference-between-flowchart-and-data.html