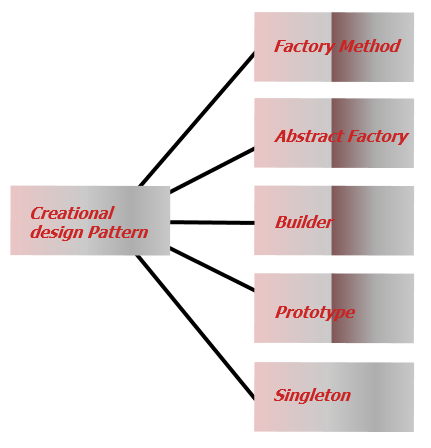
Design patterns deals with the mechanism of object creation. Basic creation may lead to lot of complication and design problem. This can be resolved with help of creational design patterns.

1. Factory pattern
2. Abstract factory patter
3. Builder pattern
4. Prototype pattern
5. Singleton pattern
6. Object pool pattern



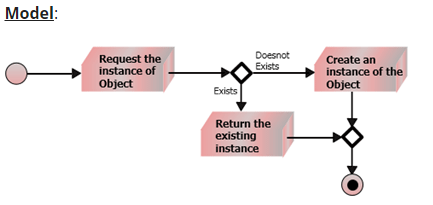
Singleton patter:

Designing a class in such a way that only one object of the class can exist

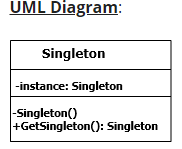
Useful for creating single global point of access to a limited resource is required. More appropriate than creating global variable. Which creates multiple point of access which lead to inconsistent value.

Pattern make sure only single object is created and ensures always there will be a single instance of the object.

Model



Uml:



Implementation:

1.class must have private static variable of singleton class

2. Implement private constructor (so that outsiders ma not instantiate it)

3. Public static method get the instance on the global object

This will act as single entry point.

Use of single ton class

Logger files,

Db connections

Driver access

Cache- memory

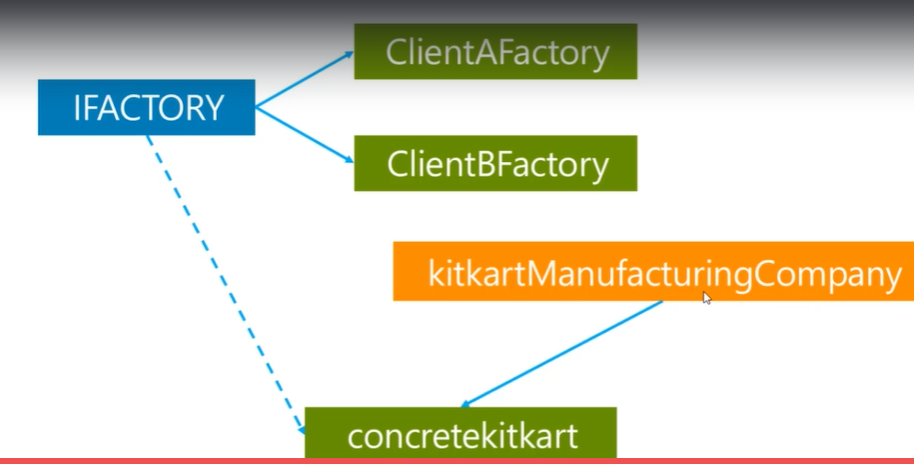
1. Hardware interface access Eg: Printer where in you don’t want multiple ppl to lock the hardware
2. Logger: Log class used for making log files where in it the logger takes data from the user and generate log if the ,multiple instance are trying to create log at same time there will be an issue they may end up editing same thing again and again
3. Cache object : Single global point of contact for all the future calls to the cache.
4. Configuration file object: Single instance is created and configuration information is maintained in memory so the next time onwards client can access information from in memory.

Factory patterns:

Real world factor is a place which produces the family of things

In code factory is a class that produces the family of the objects and we can have factory of objects and factory of methods.

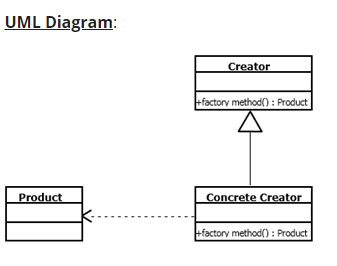
Eg: manufacturing company KitKart produces of the computer for their client multiple client



Model:



UML



Factory method takes care of one single product

Abstract factory class takes care of encapsulation of family of products.

Factory pattern:

Create concrete class instance without specifying exact class type

It is an object creation pattern

Used when sub class are given with privilege of instantiating object

Object creational pattern

We create a object without exposing the logic to create the object

Client uses same interface to create object

Eg

1. Drawing tool used to draw different shapes here the user selects different shapes from the UI In such cases we can internally use factory method to return the object of different shapes
2. Say make my trip app We can book tickets for bus, train, flight so her e what we can do is when user makes a selection from the UI for mode of transport we set values like train, bus, flight based on the travel type we return the object to book tickets from respective mode.
3. Implement abstract class called any type travel with get object method that will return the object to be used based on the type and use other functions like boarding point drop point, time etc.

Abstract Factory pattern:

Super factory that creates other factories.

Factory of factories

Interface is responsible for creating the set of related objects or dependent objects without specifying their concrete implementation.

Intermediate interface linked to another factory

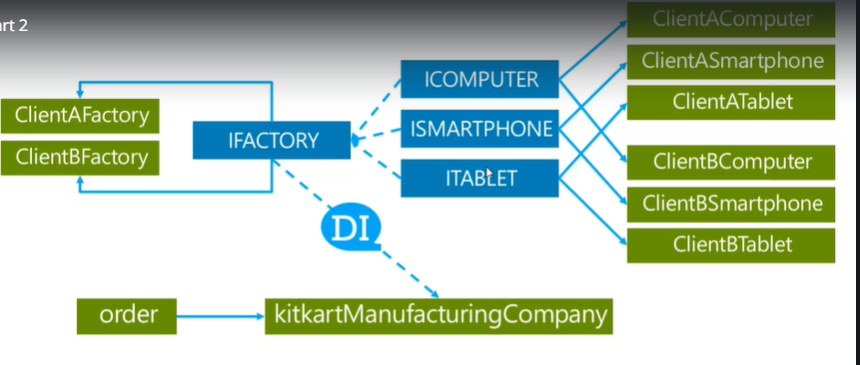
Kit kart company makes Multiple products

Computer

Mobile

Table

For multiple company

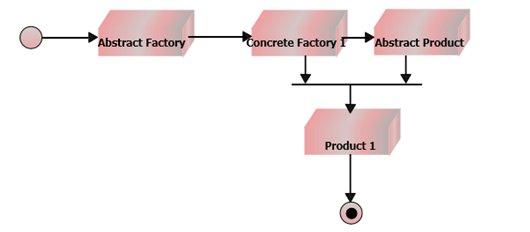


Provides the interface for creating the family of related or depended objects without specifying their concrete class

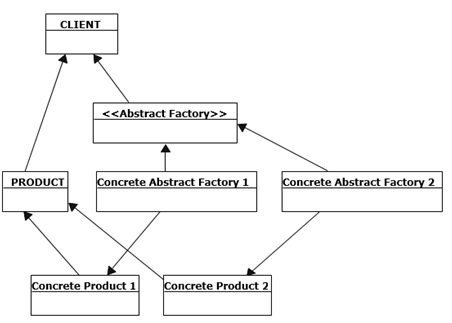
2 types

1. Abstract class defining factory method to answer the instance of the concert subclass class
2. Abstract class defining the common protocol for the factory methods.
3. Object is created without exposing the object creation logic to the client and the client use some interface to create objects

Model



UML



Builder Patter:

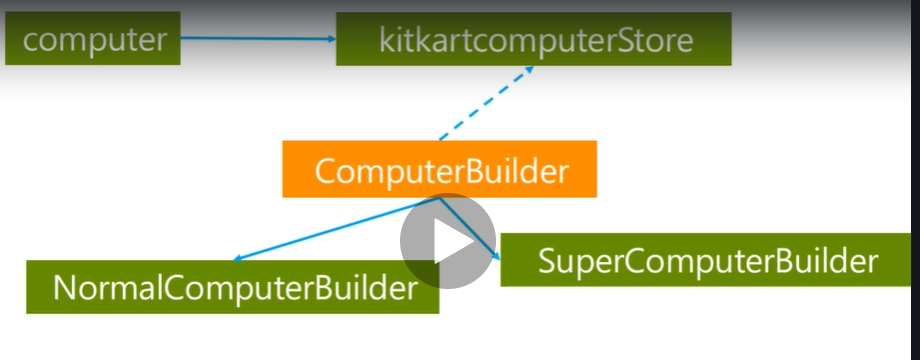
1. Used to build complex object from its representation.
2. Separate the construction of complex object.
3. So the same construction can be used build different representation.
4. Separate construction from representation.
5. Step by step process to build object

Eg: StrinBuilder used in C#

Complex string is built in steps

Eg: Kit kart company produces computer

1. Simple computer.
2. Super computer.



Here we build the object in step by step basis not as single object and basically use methods to build the class components.

Prototype pattern:

1.It act as prototype for the creation of the object

2. Using this design pattern we can create a copy/clone of the object from another existing object.

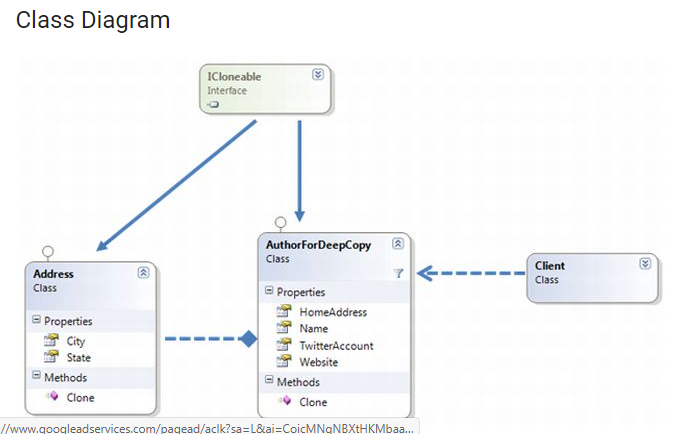
Types of the Copy:

1. Deep copy/clone
2. Shallow copy/clone

Shallow copy: used to copy all the object and reference types. But this copy won’t copy the object for which reference are referring to. The references i the new objects will still point towards the same references where the parent is pointing. (only parent objects are cloned same references will be copied)

In deep copy: Clone operation will make a clone of both parent child/ reference objects directly or indirectly.

Class diagram:



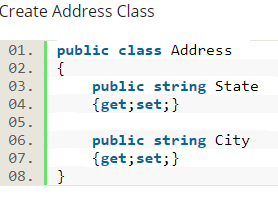
To implement this design patterns either

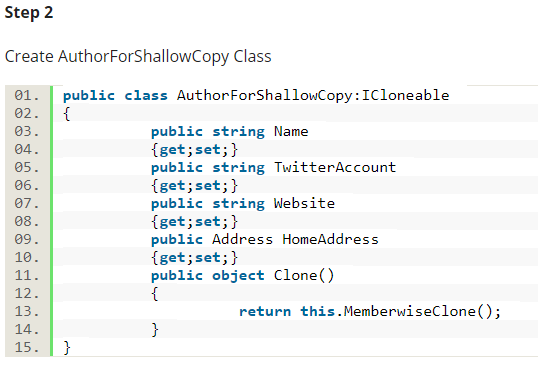
1.you have to implement IClonable interface

2. Implement Clone method from the interface

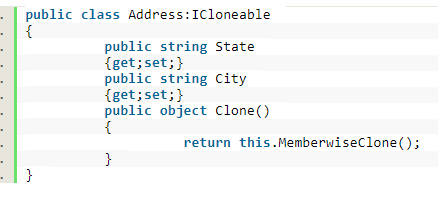
3. Use MemberWiseClone Operation on the current object and return the clone

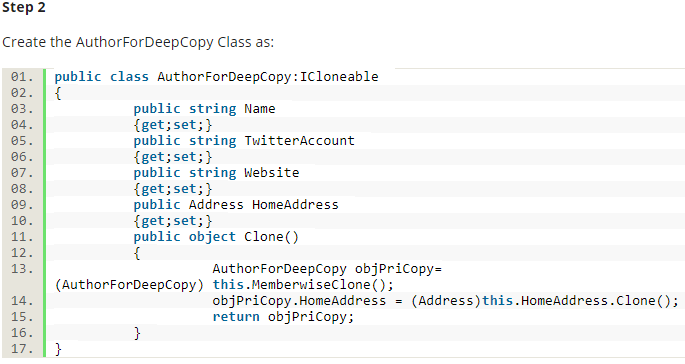
Shallow copy:





Deep copy:





In deep copy all the classes including containing class have to implement inclinable interface and implement clone methods

SO that the MemberwiseClone can be called on the each objects.

UML notation:

