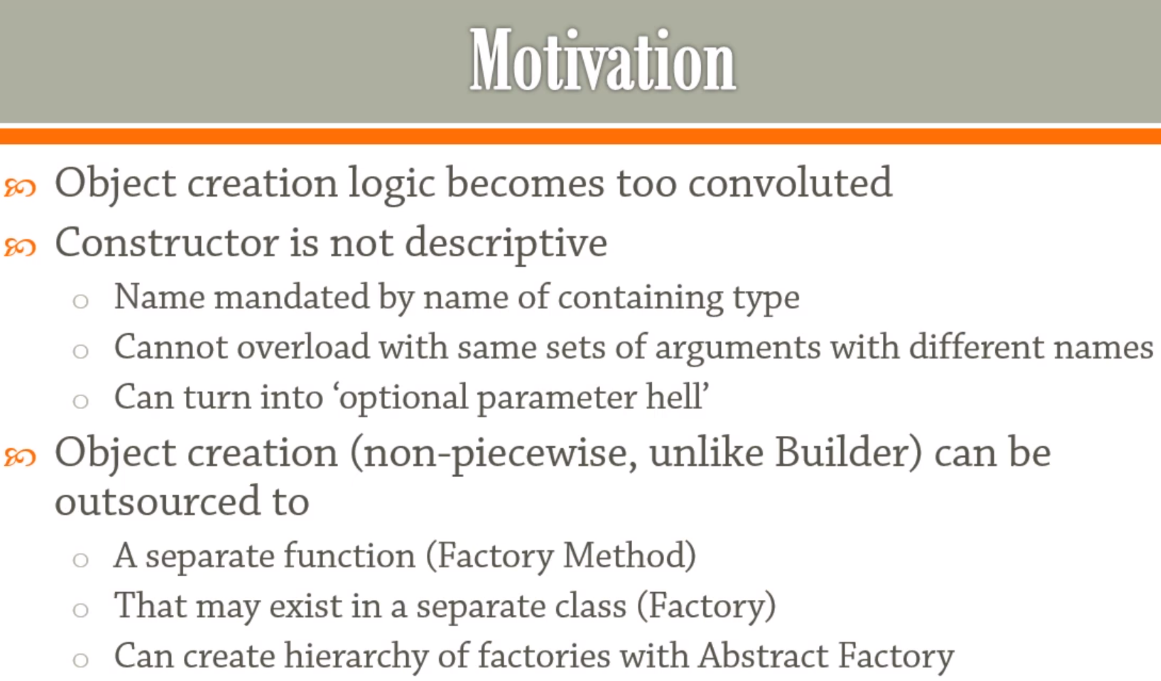
**Lesson01 What is the Factory Design Pattern**

**Notes: -**

**1-we will cover factory method and abstract factory**

**2-with factory design pattern we can implement object creation through factory method or create multiple builder and inherit from abstract factory class**



**3-with factory design pattern it will resolve the issue of the overloaded constructor of the same parameter type and name**

**(like handle point initialization between the Cartesian or Polar system ways)**

**Example of why we using factory pattern**

**public class Point{**

**private double x, y;**

**/// <summary>**

**/// Initialize point from either cartesian or polar**

**/// </summary>**

**/// <param name="a">x if cartesian , rho if polar</param>**

**/// <param name="b">y if cartesian , rho is polar</param>**

**/// <param name="system"></param>**

**public Point(double a, double b, CoordinateSystem system = CoordinateSystem.Cartesian){**

**switch (system){**

**case CoordinateSystem.Cartesian:**

**x = a;**

**y = b;**

**break;**

**case CoordinateSystem.Polar:**

**x = a \* Math.Cos(b);**

**y = a \* Math.Sin(b);**

**break;**

**default:**

**throw new ArgumentOutOfRangeException(nameof(system), system, null);}}}**

**Lesson02 method factory pattern**

**Notes: -**

**1-with factory method we define static method for each system to initialize the properties and use it on the main class as below**

**(it will resolve the issue of multiple system of point instance)**

**public class Point{**

**private double x, y;**

**public Point(double x, double y){**

**this.x = x;**

**this.y = y;}**

**//each method initialize new instance of Point**

**public static Point NewCartesianPoint(double x, double y){**

**return new Point(x, y);}**

**//each method initialize new instance of Point**

**public static Point NewPolarPoint(double rho, double theta){**

**return new Point(rho \* Math.Cos(theta), rho \* Math.Sin(theta));}**

**public override string ToString(){return $"X : {x} , Y : {y}";}}**

**//on Main method we initialize the instance directly through static method as below**

**//this is called factory builder pattern**

**class Program{**

**static void Main(string[] args){**

**var point = Point.NewPolarPoint(1.0, Math.PI / 2);**

**WriteLine(point.ToString());**

**var point2 = Point.NewCartesianPoint(10, 20);**

**WriteLine(point2.ToString());**

**ReadLine();}}**

**Lesson03 async method factory pattern**

**Notes:-**

**1-if you want to apply async operation inside the constructor you cannot , so you have to make async method and call it thorugh the instance itself**

**2-there are another / better way to apply method factory pattern as declare static method with call async method that return the instance itself**

**static async Task Main(string[] args){**

**//it will call the async factory method that it will return the class itself**

**await Foo.CreateAsync();}**

**public class Foo{**

**private Foo(){}**

**private async Task<Foo> InitAsync(){**

**await Task.Delay(1000);**

**return this;}**

**public static Task<Foo> CreateAsync(){**

**var result = new Foo();**

**return result.InitAsync();}}**