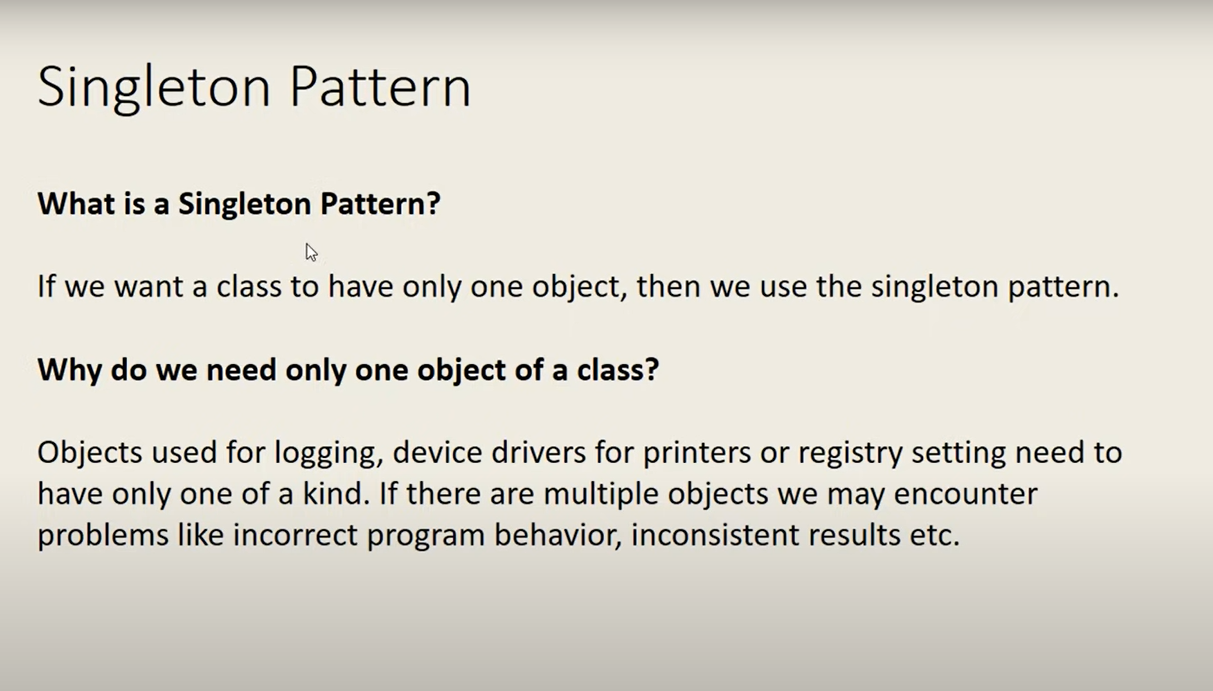




**Singleton Pattern**

**---------------------------**



Code example

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Design\_Pattern

{

public class logger

{

private static logger loggerIns;

private logger()

{

}

public static logger getInstance()

{

if(loggerIns == null)

{

loggerIns = new logger();

}

return loggerIns;

}

}

class Program

{

static void Main(string[] args)

{

logger obj1 = logger.getInstance();

logger obj2 = logger.getInstance();

Console.WriteLine(obj1.GetHashCode());

Console.WriteLine(obj2.GetHashCode());

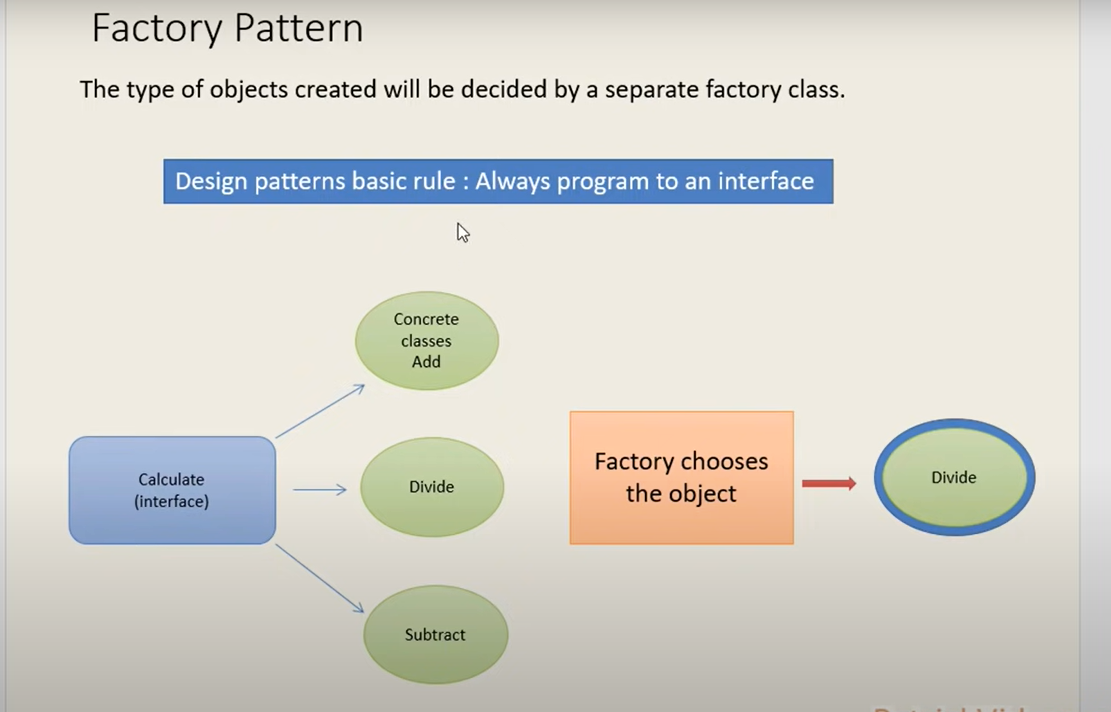
Console.Read();

}

}

}

**Factory Pattern**



* All classes must be implemented some kind of interface that makes your application flexible.

Example

-----------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Factory\_Design\_pattern

{

//interface

public interface Icalculate

{

void Calculate(double num1, double num2);

}

//implimented Class for devide

public class Divide : Icalculate

{

public void Calculate(double num1, double num2)

{

Console.WriteLine("a/b is {0}",num1/num2);

}

}

//implimented Class for Addition

public class ADD : Icalculate

{

public void Calculate(double num1, double num2)

{

Console.WriteLine("a+b is {0}", num1 + num2);

}

}

//implimented Class for substraction

public class Substract : Icalculate

{

public void Calculate(double num1, double num2)

{

Console.WriteLine("a-b is {0}", num1 - num2);

}

}

//Factory class

public class CalculateFactory

{

public Icalculate getCalculation(string Type)

{

Icalculate obj = null;

if(Type.ToLower().Equals("add"))

{

obj = new ADD();

}

else if (Type.ToLower().Equals("substract"))

{

obj = new Substract();

}

else if (Type.ToLower().Equals("devide"))

{

obj = new Divide();

}

else

{

Console.WriteLine("We don't do that");

}

return obj;

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("entyer the first no");

double num1 =Convert.ToDouble(Console.ReadLine());

Console.WriteLine("entyer the first no");

double num2 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Please enter the calculation type");

string str = Console.ReadLine();

CalculateFactory obj = new CalculateFactory();

Icalculate obj1= obj.getCalculation(str);

obj1.Calculate(num1, num2);

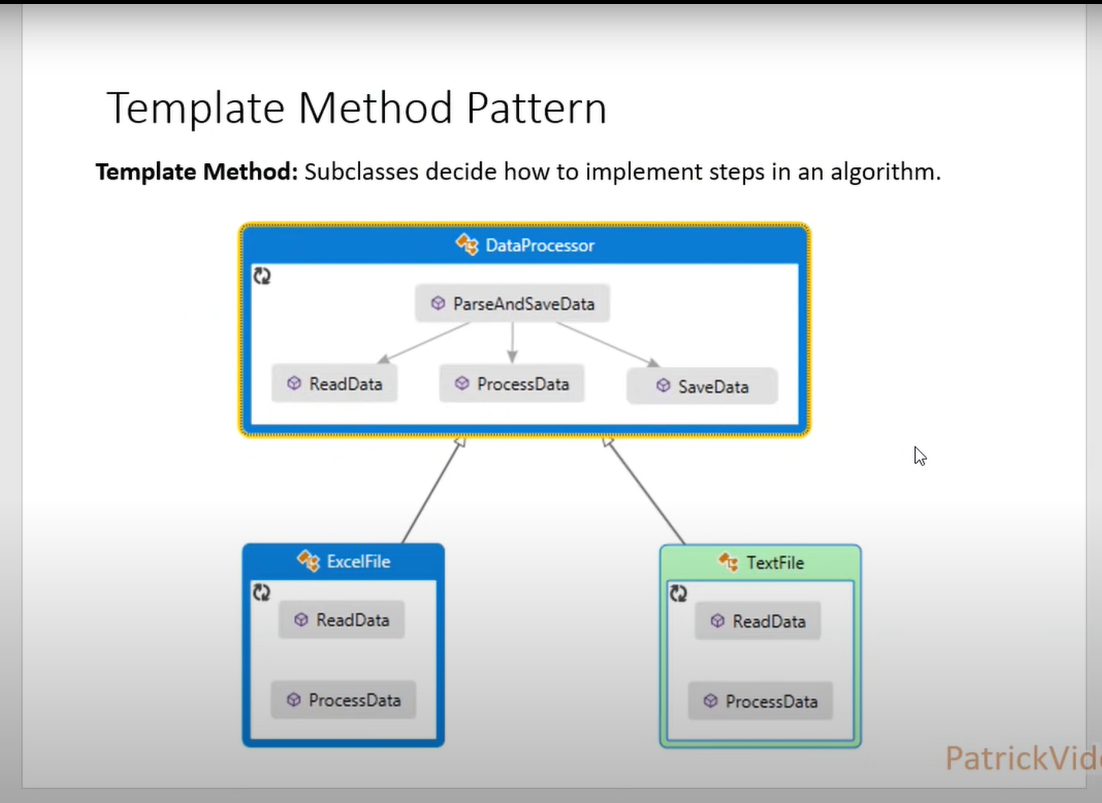
Console.Read();

}

}

}

**Template Method Pattern**

****

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Template\_design\_pattern

{

public abstract class DataProcessor

{

public void ReadProcessAndSave()

{

ReadData();

ProcessData();

SaveData();

}

public abstract void ReadData();

public abstract void ProcessData();

public void SaveData()

{

Console.WriteLine("Save data to DB");

}

}

public class ExcelFile:DataProcessor

{

public override void ReadData()

{

Console.WriteLine("Read data from excel file");

}

public override void ProcessData()

{

Console.WriteLine("process data from excel file");

}

}

public class TextFile:DataProcessor

{

public override void ReadData()

{

Console.WriteLine("Reads data from Text File");

}

public override void ProcessData()

{

Console.WriteLine("Process data from text file");

}

}

class Program

{

static void Main(string[] args)

{

DataProcessor obj = new ExcelFile();

obj.ReadProcessAndSave();

DataProcessor obj2 = new TextFile();

obj2.ReadProcessAndSave();

Console.Read();

}

}

}