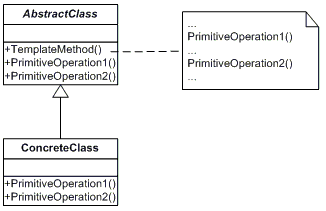
**Section21 Template Design Pattern**

**Notes: -**

**Template design pattern: Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.**



**Example: -**

**using System;**

**namespace TemplateDesignPro.Templates{**

**abstract class AbstractClass{**

**public abstract void PrimitiveOperation1();**

**public abstract void PrimitiveOperation2();**

**// The Template method group call all the functions implementations**

**public void TemplateMethod(){**

**Console.WriteLine("The Template Calling Start Here");**

**PrimitiveOperation1();**

**PrimitiveOperation2();**

**Console.WriteLine("The Template Calling End Here");}}}**

**using System;**

**namespace TemplateDesignPro.Templates{**

**class ConcreteClassA : AbstractClass{**

**public override void PrimitiveOperation1(){**

**Console.WriteLine("ConcreteClassA.PrimitiveOperation1()");}**

**public override void PrimitiveOperation2(){**

**Console.WriteLine("ConcreteClassA.PrimitiveOperation2()");}}}**

**using System;**

**namespace TemplateDesignPro.Templates{**

**class ConcreteClassB : AbstractClass{**

**public override void PrimitiveOperation1(){**

**Console.WriteLine("ConcreteClassB.PrimitiveOperation1()");}**

**public override void PrimitiveOperation2(){**

**Console.WriteLine("ConcreteClassB.PrimitiveOperation2()");}}}**

**using System;**

**using TemplateDesignPro.Templates;**

**namespace TemplateDesignPro{**

**//Template Design Pattern :**

**// Define the skeleton of an algorithm in an operation, deferring some steps to subclasses.**

**// Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.**

**class Program{**

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

**{AbstractClass aA = new ConcreteClassA();**

**aA.TemplateMethod();**

**AbstractClass aB = new ConcreteClassB();**

**aB.TemplateMethod();**

**Console.ReadKey();}}}**