Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

07

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | Implement structural design patterns (facade, decorator) on any real life system. Please make sure that you have to add some sort of innovation and logic. |
|  |  |
|  |  |
|  |  |
|  |  |

Submitted On:

\_\_\_\_\_\_\_\_\_\_\_\_

**Task No. 1:** Implement structural design patterns (facade, decorator) on any real life system. Please make sure that you have to add some sort of innovation and logic.

**Solution:**

**Façade:**

class MainApp

  {

    public static void Main()

    {

      Facade facade = new Facade();

      facade.MethodA();

      facade.MethodB();

      Console.ReadKey();}}

  class SubSystemOne

  {

    public void MethodOne()

    {

      Console.WriteLine(" SubSystemOne Method");

    }

  }

  class SubSystemTwo

  {

    public void MethodTwo()

    {

      Console.WriteLine(" SubSystemTwo Method");

    }

  }

  class SubSystemThree

  {

    public void MethodThree()

    {

      Console.WriteLine(" SubSystemThree Method");}}

  class SubSystemFour

  {

    public void MethodFour()

    {

      Console.WriteLine(" SubSystemFour Method");

    }

  }

  class Facade

  {

    private SubSystemOne \_one;

    private SubSystemTwo \_two;

    private SubSystemThree \_three;

    private SubSystemFour \_four;

    public Facade()

    {

      \_one = new SubSystemOne();

      \_two = new SubSystemTwo();

      \_three = new SubSystemThree();

      \_four = new SubSystemFour();

    }

    public void MethodA()

    {

      Console.WriteLine("\nMethodA() ---- ");

      \_one.MethodOne();

      \_two.MethodTwo();

      \_four.MethodFour();

    }

    public void MethodB()

    {

      Console.WriteLine("\nMethodB() ---- ");

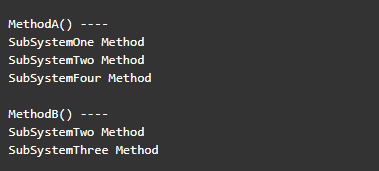
      \_two.MethodTwo();

      \_three.MethodThree();

    }

  }

**OUTPUT**



**Decorator:**

class MainApp

  {

    static void Main()

    {

      ConcreteComponent c = new ConcreteComponent();

      ConcreteDecoratorA d1 = new ConcreteDecoratorA();

      ConcreteDecoratorB d2 = new ConcreteDecoratorB();

      d1.SetComponent(c);

      d2.SetComponent(d1);

      d2.Operation();

      Console.ReadKey();

    }

  }

  abstract class Component

  {

    public abstract void Operation();

  }

  class ConcreteComponent : Component

  {

    public override void Operation()

    {

      Console.WriteLine("ConcreteComponent.Operation()");

    }

  }

  abstract class Decorator : Component

  {

    protected Component component;

    public void SetComponent(Component component)

    {

      this.component = component;

    }

    public override void Operation()

    {

      if (component != null)

      {

        component.Operation();

      }

    }

  }

  class ConcreteDecoratorA : Decorator

  {

    public override void Operation()

    {

      base.Operation();

      Console.WriteLine("ConcreteDecoratorA.Operation()");

    }

  }

  class ConcreteDecoratorB : Decorator

  {

    public override void Operation()

    {

      base.Operation();

      AddedBehavior();

      Console.WriteLine("ConcreteDecoratorB.Operation()");

    }

    void AddedBehavior()

    {

    }

  }

**Output:**

