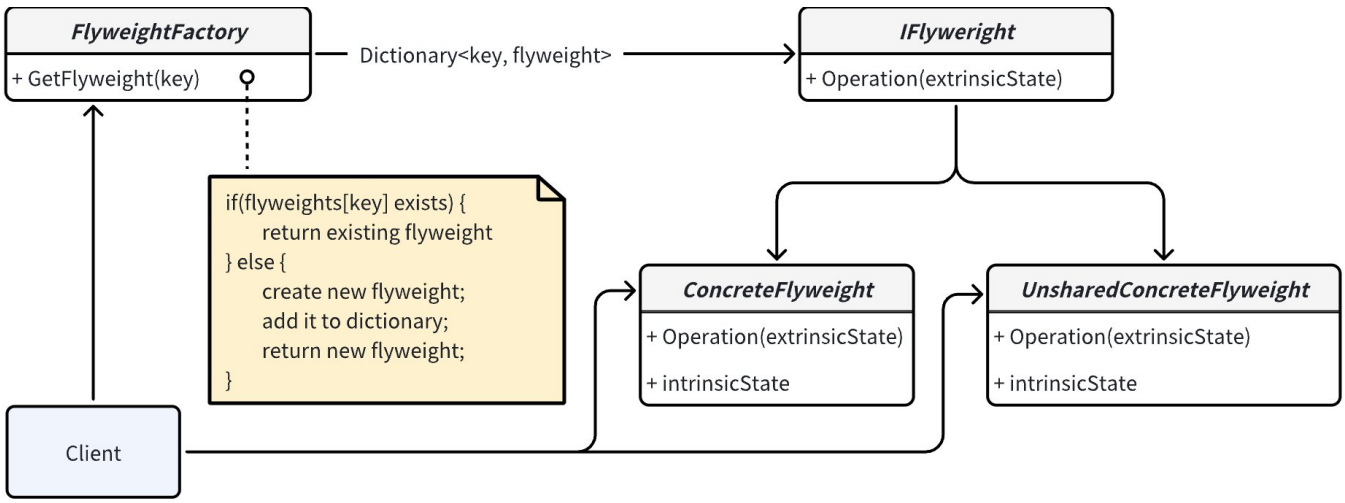
享元模式

概念

- 定义：享元模式是一种结构型设计模式，主要用于减少创建对象的数量，以减少内存占用和提高性能。它通过共享多个对象的部分状态来实现。
- 享元模式适用于以下场景：
  - 系统中有大量相似对象，这些对象会占用大量内存。
  - 对象的大部分状态可以被外部化或共享。
  - 需要通过共享来降低内存占用，如数据库连接池、字符串缓存池等。
- 享元模式主要由以下角色组成：
  - 抽象享元角色（Flyweight）：通常是一个接口或抽象类，声明了具体享元类公共的方法。
  - 享元工厂角色（Flyweight Factory）：负责创建和管理享元对象。
  - 具体享元角色（Concrete Flyweight）：实现了抽象享元类，在该类中为内部状态提供了存储空间。
  - 非共享享元角色（Unsharable Flyweight）：不能被共享的子类。
- 优点：
  - 减少内存消耗：通过共享对象，降低了系统中对象的数量，从而减少内存占用。
  - 使用了相对独立的外部状态：不会影响内部状态，能使得享元对象在不同环境中被共享。
- 缺点：
  - 享元模式需要区分外部状态和内部状态，使得应用程序某种程度上复杂化了。
  - 为了使对象可以共享，需要将享元对象的状态外部化，而读取外部状态使得运行时间变长。



实例

- 抽象享元接口：

```
public interface IShape
{
    void Draw(string size, string position); // 绘制方法，接受外部状态
}
```

- 具体享元类（可共享）：

```
public class Circle : IShape
{
    private string _color; // 内部状态：颜色

    // 构造函数接受颜色作为内部状态
    public Circle(string color)
    {
        _color = color;
    }

    // 绘制方法：接受外部状态并绘制
    public void Draw(string size, string position)
    {
        Console.WriteLine($"绘制颜色为 {_color} 的圆，大小为 {size}，位置在 {position}");
    }
}
```

- 具体享元类（非共享）：

```
public class UniqueShape : IShape
{
    private string _uniqueProperty; // 独特属性

    // 构造函数接受独特属性
    public UniqueShape(string uniqueProperty)
    {
        _uniqueProperty = uniqueProperty;
    }

    // 绘制方法：接受外部状态并绘制
    public void Draw(string size, string position)
    {
        Console.WriteLine($"绘制独特属性为 {_uniqueProperty} 的图形，
            大小为 {size}，位置在 {position}");
    }
}
```

- 享元工厂类：

```
public class ShapeFactory
{
    // 缓存已创建的享元对象
    private Dictionary<string, IShape> _shapes = new Dictionary<string, IShape>
();

    // 获取享元对象的方法
    public IShape GetCircle(string color)
    {
        if (!_shapes.ContainsKey(color))
        {
            _shapes[color] = new Circle(color); // 如果不存在该颜色的享元对象，则创
建并缓存
            Console.WriteLine($"创建了新的圆，颜色为：{color}");
        }
        return _shapes[color]; // 返回缓存中的享元对象
    }

    // 获取非共享享元对象的方法
    public IShape GetUniqueShape(string uniqueProperty)
    {
        return new UniqueShape(uniqueProperty); // 每次都创建新的非共享享元对象
```

```
    }
}
```

- 客户端代码

```
public static void Main(string[] args)
{
    ShapeFactory factory = new ShapeFactory();

    // 获取共享的圆对象，并传递外部状态
    IShape redCircle1 = factory.GetCircle("红色");
    redCircle1.Draw("10", "左上角");

    IShape redCircle2 = factory.GetCircle("红色");
    redCircle2.Draw("15", "右下角");

    IShape blueCircle = factory.GetCircle("蓝色");
    blueCircle.Draw("20", "中心");

    // 获取非共享的图形对象
    IShape uniqueShape1 = factory.GetUniqueShape("独特属性1");
    uniqueShape1.Draw("30", "左下角");

    IShape uniqueShape2 = factory.GetUniqueShape("独特属性2");
    uniqueShape2.Draw("40", "右上角");

    // 通过工厂获取的“红色”圆是同一个对象
    Console.WriteLine(Object.ReferenceEquals(redCircle1, redCircle2) ?
        "同一个对象" : "不同对象");

    // 非共享享元对象是不同的对象
    Console.WriteLine(Object.ReferenceEquals(uniqueShape1, uniqueShape2) ?
        "同一个对象" : "不同对象");
}
```

- 打印结果

```
创建了新的圆，颜色为：红色
绘制颜色为 红色 的圆，大小为 10，位置在 左上角
绘制颜色为 红色 的圆，大小为 15，位置在 右下角
创建了新的圆，颜色为：蓝色
绘制颜色为 蓝色 的圆，大小为 20，位置在 中心
绘制独特属性为 独特属性1 的图形，大小为 30，位置在 左下角
```

```
绘制独特属性为 独特属性2 的图形，大小为 40，位置在 右上角
同一个对象
不同对象
```