

原型模式

概念


- 原型（Prototype）设计模式是一种创建型设计模式，通过复制现有对象来创建新对象，而不是通过类构造函数实例化对象。
- 设计意图：
 - 避免重复的初始化代码。
 - 提高对象创建的效率。
- 使用场景
 - 当一个系统应该独立于其产品创建、构成和表示。
 - 当要实例化的类是在运行时动态创建。
 - 为了避免创建一个与产品类层次平行的工厂类层次。
 - 当一个类的实例只能有几个不同状态组合中的一种。
- 构成
 - Prototype（原型接口）：声明一个克隆自身的接口。
 - ConcretePrototype（具体原型类）：实现一个克隆自身的操作。
- 优点
 - 性能提升：通过克隆对象来创建新对象，比通过构造函数创建对象更高效。
 - 简化对象创建：避免了复杂的初始化过程。
 - 动态扩展：可以在运行时动态地增加对象。
- 缺点
 - 需要区分浅拷贝和深拷贝：对于引用类型的字段，需要实现深拷贝。
 - 克隆方法的实现：每个类都需要实现克隆方法，增加了代码复杂性。

实例

- 原型接口

```
1 public abstract class Prototype
2 {
3     // 通过deepCopy参数来决定是否需要深拷贝
4     public abstract Prototype Clone(bool deepCopy);
```

```
5 }
```

 浅拷贝：会复制对象的所有字段，但对于引用类型字段只复制引用，而不复制引用的对象本身；在C#中，可以使用 MemberwiseClone 方法实现浅拷贝。

深拷贝：不仅复制对象的所有字段，还会递归地复制引用类型字段所引用的对象，确保新对象完全独立于原始对象。

- IdInfo类；其类对象作为ConcretePrototype类中引用类型的变量

```
1 public class IdInfo
2 {
3     public int IdNumber { get; set; }
4
5     public IdInfo(int idNumber)
6     {
7         IdNumber = idNumber;
8     }
9 }
```

- 具体原型类

```
1 public class ConcretePrototype : Prototype
2 {
3     public string Name { get; set; }
4     public IdInfo IdInfo { get; set; }
5
6     public ConcretePrototype(string name, IdInfo idInfo)
7     {
8         Name = name;
9         IdInfo = idInfo;
10    }
11
12    // 实现Clone方法，支持浅拷贝和深拷贝
13    public override Prototype Clone(bool deepCopy)
14    {
15        if (deepCopy)
16        {
17            // 深拷贝，直接实例化一个新的对象
18            return new ConcretePrototype(Name, new IdInfo(IdInfo.IdNumber));
19        }
20        else
```

```

21     {
22         // 浅拷贝, 使用MemberwiseClone()方法
23         return (Prototype)this.MemberwiseClone();
24     }
25 }
26
27 public void Display()
28 {
29     Console.WriteLine($"Name: {Name}, IdNumber: {IdInfo.IdNumber}");
30 }
31 }

```

- 实例

```

1 // 创建原型对象
2 ConcretePrototype prototype1 = new ConcretePrototype("Prototype 1", new
    IdInfo(123));
3 prototype1.Display();
4
5 // 浅拷贝原型对象
6 ConcretePrototype shallowCopy = (ConcretePrototype)prototype1.Clone(false);
7 shallowCopy.Name = "Shallow Copy";
8 shallowCopy.IdInfo.IdNumber = 456;
9 shallowCopy.Display();
10
11 // 深拷贝原型对象
12 ConcretePrototype deepCopy = (ConcretePrototype)prototype1.Clone(true);
13 deepCopy.Name = "Deep Copy";
14 deepCopy.IdInfo.IdNumber = 789;
15 deepCopy.Display();
16
17 // 原型对象仍然保持不变
18 prototype1.Display();

```

- 打印结果

```

1 Name: Prototype 1, IdNumber: 123
2 Name: Shallow Copy, IdNumber: 456
3 Name: Deep Copy, IdNumber: 789
4 Name: Prototype 1, IdNumber: 456 // 浅拷贝IdInfo, 修改会影响prototype1

```