C# Design Patterns: Prototype

IMPLEMENTING THE PROTOTYPE PATTERN



Harrison Ferrone SOFTWARE DEVELOPER

@journeyman_programmer www.paradigmshiftdev.com

Summary

The Prototype Pattern in Practice:

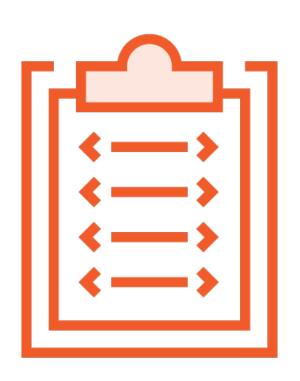
- Defining an object class
- Creating an abstract prototype class
- Handling shallow and deep copying
- Using a prototype manager
- Review use cases/practical applications

Design Pattern Categories

Creational Structural Behavioral

"Specifies the kinds of objects to create using a prototypical instance, and creates new objects by copying this prototype."

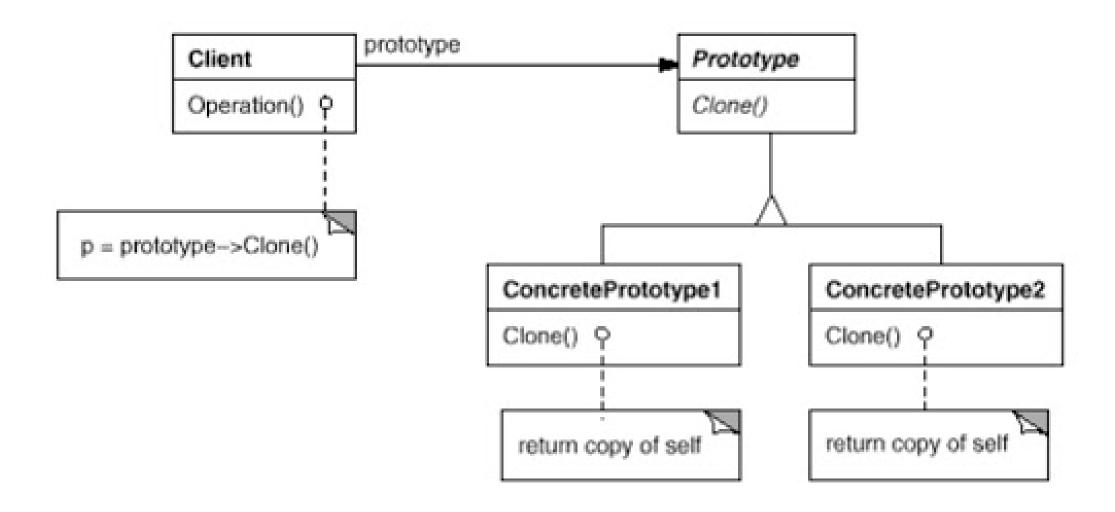
- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*



Client requests clone of an existing object

- Cloning logic is in the object itself
- Cuts down on subclassing
- Hides cloning implementation

Prototype Pattern Diagram



When object creation, composition, and representation needs to be separate from a given system

Creating an Order Class

Implementing Object Copying

Adding a Prototype Manager



Prototype works well with Object Pool

- Especially if manager class is used

Works with indefinite variety of prototypes

- As long as they all inherit from Prototype

Use Cases and Implications

Prototype Use Cases



Not for everything - all classes don't need copy behaviors



When classes to instantiate are specified at run time

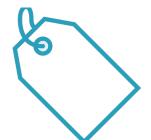


Avoiding a class hierarchy of factories that mirrors the class hierarchy of your products



When class instances can only have a small, finite combination of states

Design Pattern Implications



Concrete prototype classes are hidden from the client code



Products can be added or removed at run time by the client



Using cloned objects with varying configurations means your system will be more dynamic through object composition



Prototype objects can be used as parts in a larger object creation design, which increases modularity

Summary

How to create an abstract Prototype class

How to distinguish and implement shallow vs deep copying

How to use a prototype manager class

How to identify use cases and correct implications

Happy Coding!