Родственные паттерны

С помощью паттерна одиночка могут быть реализованы многие паттерны. См. описание абстрактной фабрики, строителя и прототипа.

Обсуждение порождающих паттернов

Есть два наиболее распространенных способа параметризовать систему классами создаваемых ей объектов. Первый способ – порождение подклассов от класса, создающего объекты. Он соответствует паттерну фабричный метод. Основной недостаток метода: требуется создавать новый подкласс лишь для того, чтобы изменить класс продукта. И таких изменений может быть очень много. Например, если создатель продукта сам создается фабричным методом, то придется замещать и создателя тоже.

Другой способ параметризации системы в большей степени основан на композиции объектов. Вы определяете объект, которому известно о классах объектов-продуктов, и делаете его параметром системы. Это ключевой аспект таких паттернов, как абстрактная фабрика, строитель и прототип. Для всех трех характерно создание «фабричного объекта», который изготавливает продукты. В абстрактной фабрике фабричный объект производит объекты разных классов. Фабричный объект строителя постепенно создает сложный продукт, следуя специальному протоколу. Фабричный объект прототипа изготавливает продукт путем копирования объекта-прототипа. В последнем случае фабричный объект и прототип – это одно и то же, поскольку именно прототип отвечает за возврат продукта.

Рассмотрим каркас графических редакторов, описанный при обсуждении паттерна прототип. Есть несколько способов параметризовать класс `GraphicTool` классом продукта:

- применить паттерн фабричный метод. Тогда для каждого подкласса класса `Graphic` в палитре будет создан свой подкласс `GraphicTool`. В классе `GraphicTool` будет присутствовать операция `NewGraphic`, переопределяемая каждым подклассом;
- использовать паттерн абстрактная фабрика. Возникнет иерархия классов `GraphicsFactories`, по одной для каждого подкласса `Graphic`. В этом случае каждая фабрика создает только один продукт: `CircleFactory` – окружности `Circle`, `LineFactory` – отрезки `Line` и т.д. `GraphicTool` параметризуется фабрикой для создания подходящих графических объектов;
- применить паттерн прототип. Тогда в каждом подклассе `Graphic` будет реализована операция `Clone`, а `GraphicTool` параметризуется прототипом создаваемого графического объекта.

Выбор паттерна зависит от многих факторов. В нашем примере каркаса графических редакторов, на первый взгляд, проще всего воспользоваться фабричным методом. Определить новый подкласс `GraphicTool` легко, а экземпляры `GraphicTool` создаются только в момент определения палитры. Основной недостаток такого подхода заключается в комбинаторном росте числа подклассов `GraphicTool`, причем все они почти ничего не делают.

Абстрактная фабрика лишь немногим лучше, поскольку требует создания равновеликой иерархии классов `GraphicsFactory`. Абстрактную фабрику следует предпочесть фабричному методу лишь тогда, когда уже и так существует иерархия класса `GraphicsFactory`: либо потому, что ее автоматически строит компилятор (как в Smalltalk или Objective C), либо она необходима для другой части системы.

Очевидно, целям каркаса графических редакторов лучше всего отвечает паттерн прототип, поскольку для его применения требуется лишь реализовать операцию `Clone` в каждом классе `Graphics`. Это сокращает число подклассов, а `Clone` можно с пользой применить и для решения других задач – например, для реализации пункта меню **Duplicate** (дублировать), – а не только для инстанцирования.

В случае применения паттерна фабричный метод проект в большей степени поддается настройке и оказывается лишь немногим более сложным. Другие паттерны нуждаются в создании новых классов, а фабричный метод – только в создании одной новой операции. Часто этот паттерн рассматривается как стандартный способ создания объектов, но вряд ли его стоит рекомендовать в ситуации, когда инстанцируемый класс никогда не изменяется или когда инстанцирование выполняется внутри операции, которую легко можно заместить в подклассах (например, во время инициализации).

Проекты, в которых используются паттерны абстрактная фабрика, прототип или строитель, оказываются еще более гибкими, чем те, где применяется фабричный метод, но за это приходится платить повышенной сложностью. Часто в начале работы над проектом за основу берется фабричный метод, а позже, когда проектировщик обнаруживает, что решение получается недостаточно гибким, он выбирает другие паттерны. Владение разными паттернами проектирования открывает перед вами широкий выбор при оценке различных критериев.