# Часть 2. Порождающие паттерны

В большинстве объектно-ориентированных языков программирования за конструирование объекта отвечает конструктор. В самом простом случае, клиент знает, какой класс объекта ему требуется и в нужный момент времени вызывает один из доступных конструкторов. Но наличие наследования или более сложный процесс создания может сделать использование конструктора неудобным или даже невозможным. Порождающие паттерны предназначены для решения типовых проблем, которые возникают при создании объектов.

Паттерн *Синглтон* по праву считается самым известным паттерном проектирования (\*). Он гарантирует, что будет создан лишь один экземпляр класса и обеспечивает глобальную точку доступа к этому объекту. Последний аспект синглтонов делает его похожим на глобальную переменную, чрезмерное использование которой приводит к хрупкому дизайну и возрастающей сложности.

(\*) Чрезмерное использование этого паттерна часто приводит к большому числу неявных связей в приложении, что существенно усложняет его сопровождаемость. Именно поэтому многие считают Синглтон не столько паттерном, сколько \*анти-паттерном\* проектирования.

Вторым по популярности паттерном проектирования является *Фабрика*. Существует несколько разновидностей фабрик. *Абстрактная фабрика* предназначена для создания семейства объектов и позволяет заменять это семейство путем использования нужного подкласса фабрики. Статический фабричный метод представляет собой статический метод, который возвращает экземпляр конкретного класса или класса из иерархии наследования, в зависимости от аргументов метода или конфигурации. Классический фабричный метод создает экземпляр полиморфного типа и обеспечивает расширяемость путем создания и использования наследования.

Если процесс создания сложный, но отвечать за него должен клиент, то вместо фабрики следует использовать паттерн *Строитель*. Строитель позволяет клиентам собирать сложный объект по кусочкам, не вдаваясь в подробности того, как из этих кусочков получается окончательный результат. Строитель отлично подходит для создания тестируемых классов и тестовых данных, но активно применяется и в логике приложения.