Pattern	Область применения	Преимущества	Недостатки
Singleton	Часто в системе могут существовать сущности только в единственном экземпляре, например, система ведения системного журнала сообщений или драйвер дисплея. В таких случаях необходимо уметь создавать единственный экземпляр некоторого типа, предоставлять к нему доступ извне и запрещать создание нескольких экземпляров того же типа.	• Класс сам контролирует процесс создания единственного экземпляра. • Паттерн легко адаптировать для создания нужного числа экземпляров. • Возможность создания объектов классов, производных от Singleton.	• В случае использования нескольких взаимозависимых одиночек их реализация может резко усложниться.
Abstract Factory	Используется если: система должна оставаться независимой как от процесса создания новых объектов, так и от типов порождаемых объектов или необходимо создавать группы или семейства взаимосвязанных объектов, исключая возможность одновременного использования объектов из разных семейств в одном контексте.	• Скрывает сам процесс порождения объектов, а также делает систему независимой от типов создаваемых объектов, специфичных для различных семейств или групп. • Позволяет быстро настраивать систему на нужное семейство создаваемых объектов.	• Трудно добавлять новые типы создаваемых продуктов или заменять существующие, так как интерфейс базового класса абстрактной фабрики фиксирован.
Builder	Используется в случае, когда в системе могут существовать сложные объекты, создание которых за одну операцию затруднительно или невозможно. Требуется поэтапное построение объектов с контролем результатов выполнения каждого этапа.	• Возможность контролировать процесс создания сложного продукта. • Возможность получения разных представлений некоторых данных. • Изолирует код, реализующий конструирование и представление;	• ConcreteBuilder и создаваемый им продукт жестко связаны между собой, поэтому при внесении изменений в класс продукта скорее всего придется соответствующим образом изменять и класс ConcreteBuilder.
Strategy	Используется в системах, поведение которых может определяться согласно одному алгоритму из некоторого семейства. Все алгоритмы этого семейства являются родственными: предназначены для решения общих задач, имеют одинаковый интерфейс для использования и отличаются только реализацией (поведением). Пользователь, предварительно настроив программу на нужный алгоритм, получает ожидаемый результат.	• Систему проще поддерживать и модифицировать, так как семейство алгоритмов перенесено в отдельную иерархию классов. • Паттерн Strategy предоставляет возможность замены одного алгоритма другим в процессе выполнения программы. • Паттерн Strategy позволяет скрыть детали реализации алгоритмов от клиента.	• Для правильной настройки системы пользователь должен знать об особенностях всех алгоритмов. • Число классов в системе, построенной с применением паттерна Strategy, возрастает.