Стратегия(Strategy)

Стратегия используется, когда вы хотите расширить поведение объекта, где это поведение может изменяться во время выполнения. Если нескольким объектам нужно использовать одно и то же поведение (алгоритм), мы также получаем преимущество повторного использования кода.

Внедрение зависимости(DI)

Зависимость используется, когда вы хотите разделить зависимости объекта и передать их зависимому объекту во время выполнения. Зависимый объект не должен знать, как построить зависимости, и какие фактические зависимости он использует.

State(Государственный)

Поведение класса может изменяться в зависимости от набора состояний, созданных пользователем или внутри системы. В этом паттерне мы инкапсулируем каждое состояние. Пользователю не нужно знать о каждом состоянии, пользователь выполняет только некоторые действия, которые, в свою очередь, могут изменить состояние объекта.

Адаптер

Адаптер помогает объединить два несовместимых интерфейса для совместной работы. Итак, если у вас есть интерфейс с реализующими классами. Если позже вас попросили добавить дополнительные подклассы, но у них несовместимый интерфейс, тогда может пригодиться шаблон адаптера. Есть две структуры:

декоратор

Шаблон декоратора динамически расширяет функциональность объекта.

Мост

Разъединяет абстракцию от ее реализации, так что они могут различаться независимо друг от друга. Например, если у вас есть класс с именем Rectangle. Этот класс может иметь две разные реализации, Red Rectangle и Blue. Вместо наследования от класса Rectangle, одного для синего прямоугольника и другого для красного, мы могли бы вместо этого извлечь эти реализации и использовать Composition over Inheritance.

Композитный(Composite)

Используется для создания древовидной структуры группы объектов. Таким образом, объект может быть набором других объектов, где объекты имеют общий интерфейс, который определяет общие операции.

Объект может иметь коллекцию объектов с именем Composite Or Node, в то время как объекты, которые не могут иметь другие объекты (на самом низком уровне), называются Leaf. Композитный объект может иметь листья или другие композиты.

Итератор(Iterator)

Этот шаблон используется для последовательного доступа к элементам объекта коллекции без раскрытия его базового представления. В этом фрагменте я использую встроенные в Java классы Iterable & Iterator.

Наблюдатель(Observer)

Шаблон наблюдателя используется таким образом, что, если объект изменяется, его зависимые объекты получают уведомление об этом изменении. Таким образом, существует отношение 1: М. Например, при наличии издателя, который публикует новости для подписчиков, при каждом добавлении новых обновлений или данных подписчики получают уведомление. В этом фрагменте я использую классы Java Observer и Observable.