1. **Memento Pattern – история на продукт:**

Без да се нарушава инкапсулацията, шаблонът позволява заснемане и възстановяване на вътрешното състояние на даден обект. Това дава възможност да се отменят направени промени, като се запази и възстанови състоянието на обекта до предишен момент. Полезен, при създаване на контролни точки в програмата – тук, за създаване на графика на направени промени по даден продукт.

Originator:

Отговаря за създаването и поддържането на състоянието на обекта. Директно комуникира с Memento, за да създава моментни снимки на състоянието си и да го възстановява.

Memento:

Осигурява единствено начин за извличане на състоянието, без възможност за директна модификация, което гарантира неговата цялост.

Caretaker:

Отговаря за съхраняването и управлението на обектите Memento. Не знае детайлите на състоянието, съхранено в Memento, но може да изисква от Originator да създаде или възстанови състоянието.

Client:

Инициира заявки за запазване или възстановяване на състоянието на Originator чрез Caretaker.

Grapher:

Прави графика.

1. **Strategy Pattern – смяна на езика:**

Поведенчески шаблон, който позволява да се дефинира семейство от алгоритми или поведения, които да са в отделни класове и да могат да се заменят по време на изпълнение.

Позволява се капсулиране на множество алгоритми или поведения в отделни класове, наречени стратегии, в случая езиците английски и немски. Всяка стратегия капсулира конкретно поведение. Клиентът, в случая потребителя през конзолата, може да сменя различни стратегии т.е. езиците по време на изпълнение.

Context:

Клас, наречен Контекст, възлага задачата на обект стратегия и съдържа препратка към него. Той служи като посредник между клиента и стратегията. Дефакто от него „дърпаме“ нужното съобщение, което да се изпише на конзолата.

Strategy Interface:

Определя набор от методи, които всички конкретни стратегии трябва да имплементират.

Concrete Strategies:

Те са взаимозаменяеми и могат да бъдат избирани и конфигурирани от клиента в зависимост от изискванията на задачата.

Client:

Конзолата.

Тъй като комуникацията между компонентите е decoupled, контекстът не е необходимо да знае точните детайли за това как всяка стратегия изпълнява задачата.

1. **Command Pattern – множество от команди:**

Поведенчески шаблон, който преобразува заявката в самостоятелен обект, наречен команда. С помощта на този шаблон можете да се „улови“ всеки компонент на заявката, включително обекта, който притежава метода, параметрите на метода и самия метод.

Command Interface:

Интерфейсът на командата е като наръчник, който всички класове с команди следват.

Concrete Command Classes:

Това са конкретните команди, като например създай обява, изтрий обява, регистрирай потребител и т.н.

Invoker:

Той съдържа препратка към команда, но не навлиза в подробности за това как тя работи. Като бутон – изпълни.

Receiver:

Получателят е устройството, което знае как да изпълни действителната операция, свързана с командата – конзолата, тук.

1. **Factory Pattern – от ООП:**

Предоставя интерфейс за създаване на обекти в суперклас, като позволява на подкласовете да определят типа на създаваните обекти.

Creator:

Интерфейс, който декларира „фабричния“ метод.

Concrete Creator:

Всеки конкретен създател е отговорен за създаването на определен продукт.

Product:

Дефинира общите характеристики на обектите.

Concrete Product:

Конкретен продукт.

**Coupling** се отнася до степента на взаимозависимост между софтуерните модули. High Coupling означава, че модулите са тясно свързани, което води до зависимост между тях. Промени в един модул могат да засегнат други модули. Low Coupling означава, че модулите са независими един от друг. Промени в един модул оказват минимално влияние върху останалите.

**Cohesion** се отнася до степента, в която елементите в даден модул работят заедно, за да изпълнят единна и добре дефинирана цел. High Cohesion означава, че елементите в модула са тясно свързани и се фокусират върху една конкретна отговорност. Low Cohesion означава, че елементите в модула са слабо свързани и изпълняват множество несвързани задачи, което затруднява поддръжката, разширяването и разбирането на кода.