Конспект Гришутенко Павла

1 grasp

GRASP - общие шаблоны распределения ответственностей; также существует английское слово "grasp" — «контроль, хватка») — шаблоны, используемые в объектно-ориентированном проектировании для решения общих задач по назначению ответственностей классам и объектам. [wiki]

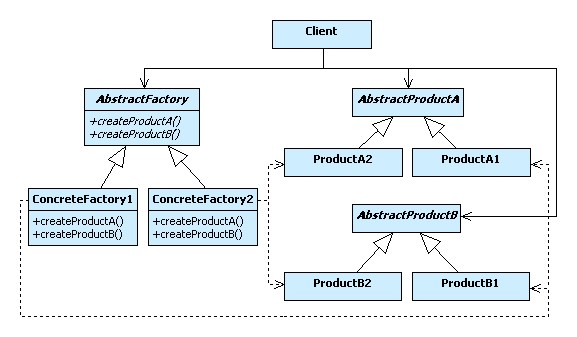
2 solid

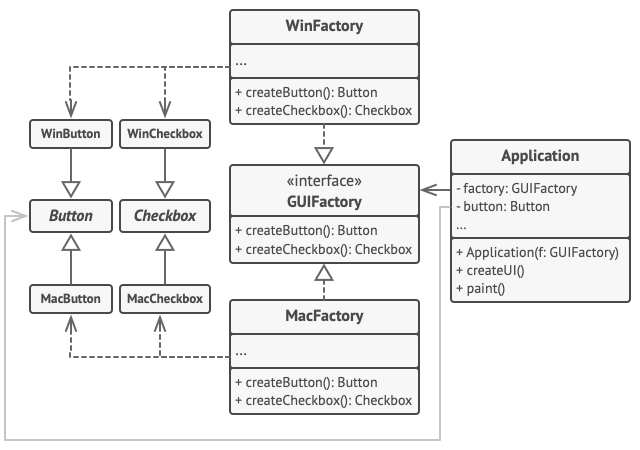
SOLID - (сокр. от англ. single responsibility, open–closed, Liskov substitution, interface segregation и dependency inversion) в программировании — мнемонический акроним, введённый Майклом Фэзерсом (Michael Feathers) для первых пяти принципов, названных Робертом Мартином в начале 2000-х, которые означали 5 основных принципов объектно-ориентированного программирования и проектирования. [wiki]

| **Инициал** | **Представляет** | **Названи, понятие** |
| --- | --- | --- |
| **S** | SRP | **Принцип единственной ответственности (single responsibility principle)**  Для каждого класса должно быть определено единственное назначение. Все ресурсы, необходимые для его осуществления, должны быть инкапсулированы в этот класс и подчинены только этой задаче. |
| **O** | OCP | **Принцип открытости/закрытости (open–closed principle)**  «программные сущности … должны быть открыты для расширения, но закрыты для модификации». |
| **L** | LSP | **Принцип подстановки Лисков (Liskov substitution principle)**  «объекты в программе должны быть заменяемыми на экземпляры их подтипов без изменения правильности выполнения программы». См. также контрактное программирование.  Наследующий класс должен дополнять, а не изменять базовый. |
| **I** | ISP | **Принцип разделения интерфейса (interface segregation principle)**  «много интерфейсов, специально предназначенных для клиентов, лучше, чем один интерфейс общего назначения». |
| **D** | DIP | **Принцип инверсии зависимостей (dependency inversion principle)**  «Зависимость на Абстракциях. Нет зависимости на что-то конкретное». |

3 patterns

Абстрактная фабрика

1. Относится к порождающим паттернам проектирования.
2. Используется, когда нужно получить семейство объектов без привязки этих объектов к конкретному классу.
3. Ошибки использования:
4. схема
5. Реализация унифицированного механизма создания gui.



источнки: <https://refactoring.guru/ru/design-patterns/abstract-factory>

Адаптер

1. Относится к структурным паттернам проектирования.
2. Задача создать обертку между несовместимыми интерфейсами, которая позволит им работать вместе.

источнки: <https://refactoring.guru/ru/design-patterns/adapter>

Компоновщик

1. Относится к структурным паттернам проектирования.
2. Позволяет компоновать объекты в древовидную структуру, а затем работать с этой структурой, как с единым объектом.

источнки: <https://refactoring.guru/ru/design-patterns/composite>

Декоратор

1. Относится к структурным паттернам проектирования.
2. Декоратор позволяет динамически добавлять объектам новую функциональность.

источнки: <https://refactoring.guru/ru/design-patterns/decorator>

Фабричный метод

1. Относится к порождающим паттернам проектирования.
2. Определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.

источнки: <https://refactoring.guru/ru/design-patterns/factory-method>

Наблюдатель

1. Относится к поведенческим паттернам проектирования.
2. Создает механизм подписки, который позволяет одним объектам следить и реагировать на изменения в других объектах.

источнки: <https://refactoring.guru/ru/design-patterns/observer>

Стратегия

1. Относится к поведенческим паттернам проектирования.
2. Помещает схожие алгоритмы в отдельные классы после чего, их можно заменять во время runtime’а.

источнки: <https://refactoring.guru/ru/design-patterns/strategy>

Шаблонный метод

1. Относится к поведенческим паттернам проектирования.
2. Определяет скелет алгоритма, перекладывая ответственность за некоторые его шаги на подклассы. Паттерн позволяет подклассам переопределять шаги алгоритма, не меняя его общей структуры.

источнки:<https://refactoring.guru/ru/design-patterns/template-method>

ЗАДАНИЕ

1. GRASP
2. SOLID
3. Шаблоны проектирования:
   * абстрактная фабрика;
   * адаптер;
   * компоновщик;
   * декоратор;
   * фабричный метод;
   * наблюдатель;
   * стратегия;
   * шаблонный метод.

При описании шаблонов рекомендуется придерживаться следующей структуры:

1. К какой категории шаблонов относится.
2. В каких случаях используется.
3. Ошибки при использовании.
4. Как графически представляется (схема).
5. Формулировка задачи, при решении которой может быть использован.
6. Решение задачи. Разбор кода решения.
7. Постановка отдельной задачи для практики использования этого паттерна.