Принципы и механизмы объектно-ориентированного проектирования

Задачи ООАП

Дизайн (результат проектирования) должен:

1) соответствовать решаемой задаче;

2) быть достаточно общим, чтобы учесть возможные изменения и дополнения, а также чтобы свести к минимуму необходимость перепроектирования;

3) создавать и применять повторно используемые артефакты.

Три кита ООП с точки зрения ООАП

1. Инкапсуляция:

Состояние объекта инкапсулировано, к нему нельзя получить непосредственный доступ, структура объекта должна быть закрыта для внешнего мира

ЕДИНСТВЕННЫЙ способ изменить внутреннее состояние объекта – послать ему запрос (выполнить его операцию).

Первый принцип объектно-ориентированного проектирования

Три кита ООП с точки зрения ООАП

2. Полиморфизм:

- Об объектах известно только то, что сообщается в их интерфейсах.
- Никакого способа получить информацию об объекте или заставить его что-то сделать в обход интерфейса просто не существует.
- Интерфейс объекта ни каким образом не определяет способ его реализации.
- Разные объекты могут обрабатывать сходные или одинаковые запросы абсолютно по разному.
- Два объекта с одинаковыми интерфейсами могут иметь абсолютно разную реализацию.

Три кита ООП с точки зрения ООАП

3. Наследование:

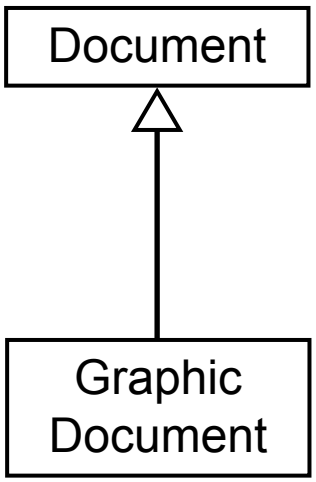
Наследование – механизм расширения функциональности приложения путем **повторного использования** функциональности родительских классов.

Четыре?

~~Три~~ кита ООП с точки зрения ООАП

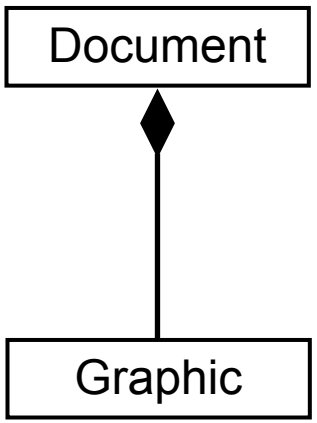
Механизмы **повторного использования**:

1) наследование;
2) композиция.



?

или



Наследование vs Композиция

- **Наследование** позволяет определить реализацию одного класса в терминах другого. Такое повторное использование называют *прозрачным или белым ящиком (white-box reuse).* Этим самым подчеркивается, что реализация предка известна потомкам.
- **Композиция**: новая функциональность получается за счет совместной работы объединения или кооперации объектов. Для композиции требуется, чтобы объекты-участники имели строго определенные интерфейсы, а детали реализации скрываются. Этот способ называется *черным ящиком (black-box reuse).*

Наследование

Достоинства и недостатки

+ Определяется статически на этапе компиляции.
+ Проще поддерживать, так как оно напрямую реализуется через язык программирования.
+ Упрощается задача модификации существующей реализации.

- Нельзя изменить унаследованную реализацию во время исполнения, так как она определяется на этапе компиляции.
- Родительский класс по крайней мере частично определяет реализацию своих подклассов. Таким образом наследование НАРУШАЕТ инкапсуляцию.

Композиция

Достоинства и недостатки

+ Доступ к объектам осуществляется только через их интерфейсы, следовательно, не нарушается инкапсуляция.
+ Во время исполнения один объект можно всегда заменить другим, если он имеет тот же тип.
+ При реализации объекта в первую очередь кодируются его интерфейсы, следовательно зависимость от реализации снижается.
+ Каждый класс в этом случае инкапсулирует только свое поведение. Классы становятся небольшими и не разрастаются до неуправляемых размеров

- Дополнительная работа по компоновке объектов.
- Требуется тщательное проектирование интерфейсов, чтобы один объект можно было использовать со множеством других.

Наследование vs Композиция

Композицию стоит предпочитать наследованию.

Второй принцип объектно-ориентированного проектирования

Паттерны проектирования

Паттерн (шаблон) проектирования – это описание взаимодействия **объектов** и **классов**, адаптированное для решения общей задачи проектирования в конкретном контексте.

Другими словами, паттерны – это некоторые обобщенные решения задач, часто встречающихся при проектировании программных приложений.

Что относится к паттернам?

- Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования.

GoF

- Ларман К. Применение UML и шаблонов проектирования (в России — более позднее издание)

GRASP

- Фаулер М. Архитектура корпоративных программных приложений.

Архитектурные решения

Главные детали паттерна

- *Имя* – ключевое слово (словосочетание), с помощью которого можно быстро описать ситуацию = проблема+решение+последствия.
- *Задача* (проблема) – описание того, когда следует применять паттерн.
- *Решение* – описание того, как проблема может быть решена в обобщенных терминах.
- *Результаты* (последствия, плюсы и минусы) – хорошее и плохое, типично возникающее в результате применения паттерна.

Неглавные детали паттерна

- Пример проблемы.
- Особенности реализации.
- Связанные паттерны: разновидности, частные случаи, паттерны, которые решают похожую задачу, альтернативные решения, применяемые или те, в которых применяется данный паттерн.

Паттерны GRASP (К. Ларман)

- файл Распределение обязанностей.ppt

Паттерны GoF (Э. Гамма и др.)

Классификация по назначению паттерна:

- порождающие – решают вопросы создания объектов;
- структурные – описывают проблемы и способы композиции объектов;
- паттерны поведения – решают проблемы взаимодействия объектов.

Классификация по уровню применимости: указывает, к чему применяется паттерн – к **классам** или **объектам**.