Tuesday, February 8, 2022 6:59 PM

Объектно-ориентированный анализ и проектирование

Диаграммы классов UML

Атрибуты — +открытый
-закрытый
статический
константный : int =10
Закрытый по умолчанию
коллекция объектов :
Bag [*]
/производный

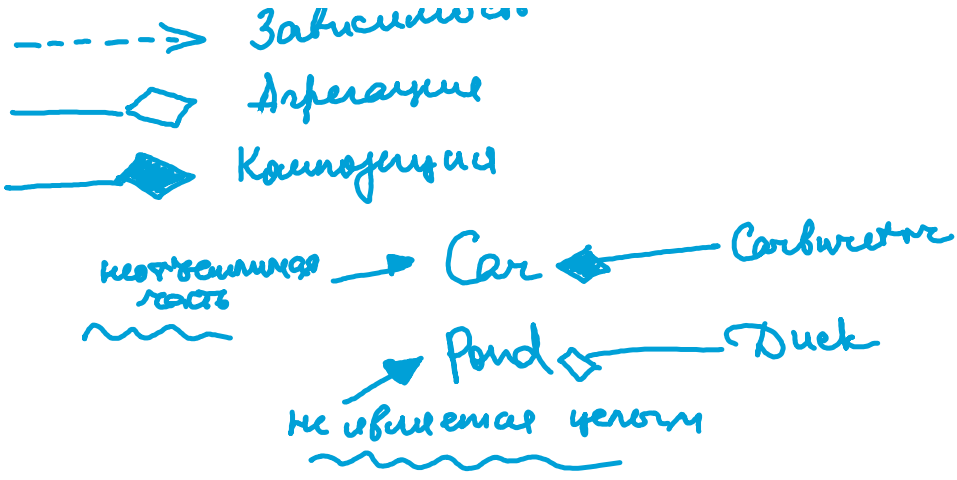
Методы:
+открытый метод ()
+стат. метод ()
открытый по умолчанию ()
-закрытый метод ()
#защищенный метод
~ метод уровня пакета ()
абстрактный метод ()
<< создать свой >> Class A()

Описание класса

Имя, свойства — атр. и ассоциации,
Операции — методы, поведение
Ограничения — условия на атрибуты,
пред- и пост- условия на атрибуты

Виды ассоциаций с навигированием

——→ Ассоциация
——▷ Наследование
- - - -▷ Реализация / имплементация



Композиция — сильное отношение

* Для зависимостей используются стереотипы

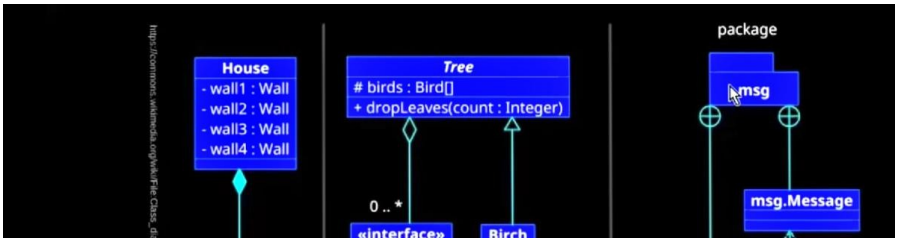
Например Car - - «use» - - > Wheel

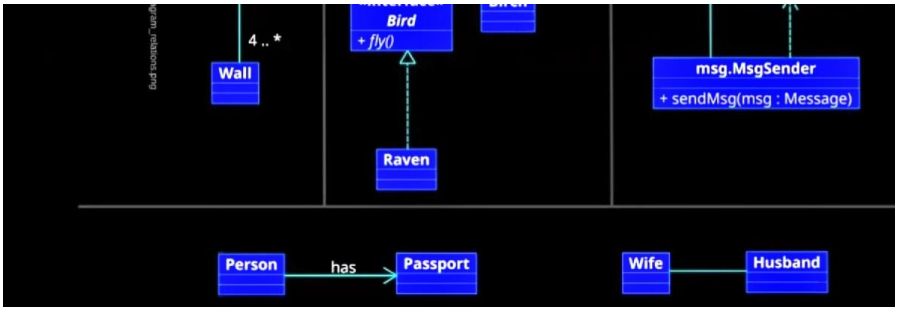
Когда лучше использовать АТРИБУТ, а когда АССОЦИАЦИЮ?

Атрибут — концептуально важный является значение (число, строка, символ)

Ассоциация — концептуально важна уникальная сущность объекта (двигатель, отчет, самолет ...), либо имеется поведение (самолет) тогда ассоциация с отдельным классом

Пример:





Абстрактные классы и объекты

У объектов иногда указываются атрибуты со значениями, но не указываются операции

⊕ Зависимости стереотипизируются ⇒ Усложняют диаграмму

Анализ и Проектирование — дисциплина унифицированного процесса

Цели 1) превратить модель требований в проект системы

2) обеспечить построение устойчивой архитектуры

3) Адаптировать проект к новым Требованиям, условиям реализации, нефункц. тр-ям

Модель предметной области

Модель проекта

Анализ

Модель проектирования

Архитектурный

Роли и артефакты ООАнП

Архитектор → модели проектирования
Архитектор → архитектурной документ

Осуществление прецедента
Пакет/подсистема
Класс
← Проектировщик

Работы ООАнП

Архитектор
- Архитектурный анализ
- Идентификация проектных механизмов
- Идентификация проектных элементов
- Описание исполнимой архитектуры
- Описание распределения
- Внедрение существующих проектных элементов

Проектировщик

Анализ прецедентов
Проектирование прецедентов
Проект-ние подсистем
Проект-ние классов

Работа с требованиями — МПО, МП, n×СП

— артефакты

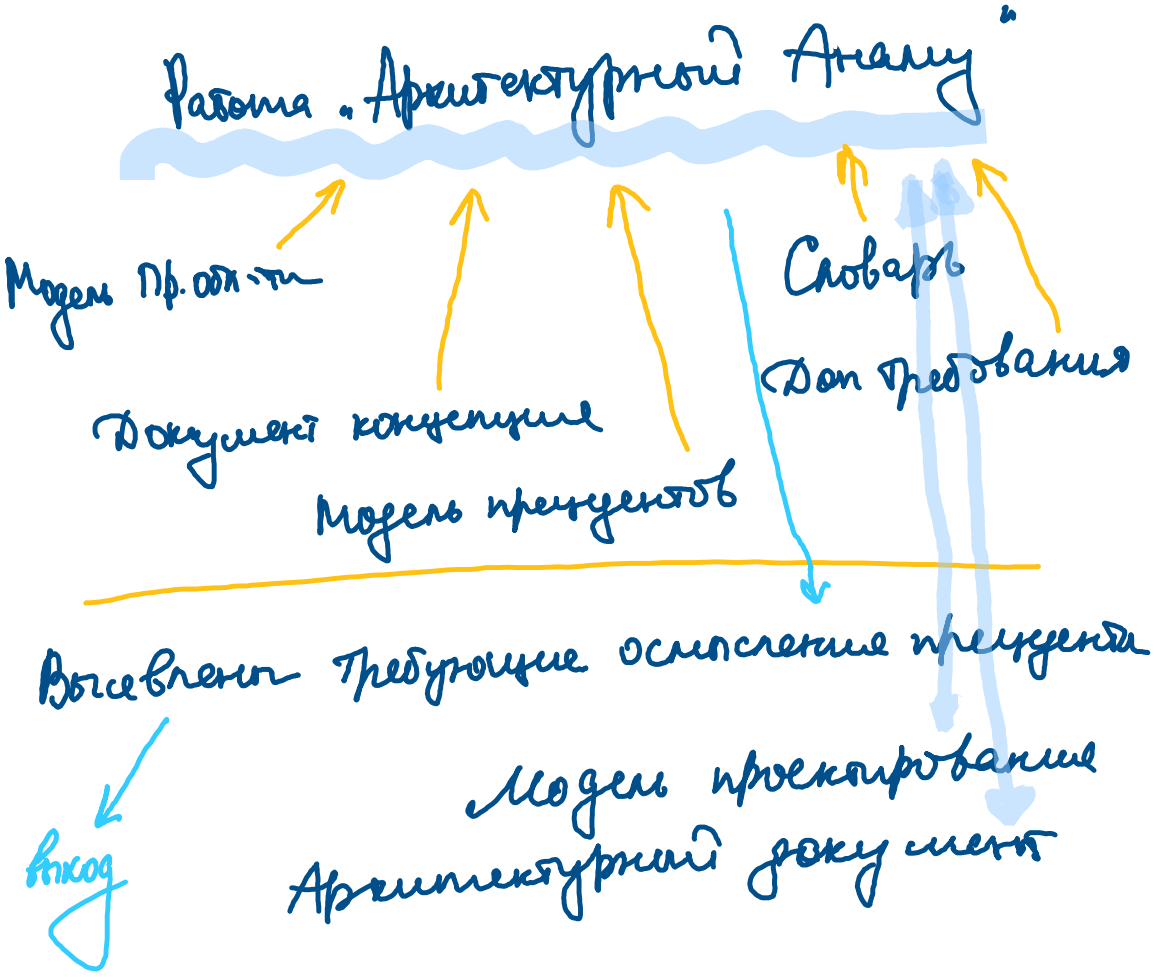
Фокус ООАиП

Анализ	Проектирование
Понять проблему Идеализированный проект Поведение объектов среды Функц. требования Небольшая высокоуровневая модель	→ Предложить решение Почти исходный код Операции, атрибуты объектов Нефункц. требования Полномасштабная детальная

Итерационный подход — архитектура системы эволюционирует в ходе итераций и постепенно стабилизи-

руется

Для поддержания целостности проекта и трассируемости требований используется концепция use-case realization (реализация прецедентов)



Шаги архитектурного анализа

1) Определить высокоуровневую организацию подсистем

2) Идентифицировать механизмы анализа

3) Идентифицировать ключевые абстракции

3) [illegible]
4) Создать реализации прецедентов

1 Архитектурные шаблоны

Архитектор определяет наиболее общую организацию системы
Шаблоны рекомендуются

Шаблон — типичное решение типичной проблемы в заданном контексте

Бывают шаблоны:
- проектирования
- анализа
- разработки
- организации разработки
- тестирования
- документирования
- построения архитектуры

Шаблоны и Каркасы

Каркас — задаёт "скелет", основу решения
↓
набор инструментов для реализации шаблонов

Пакеты UML

- универсальный механизм группировки сущностей

* Для организации проекта системы
Для организации инкапсуляции на уровне пространств имён

Для определения "семантических" границ той или иной модели в рамках проекта

Для организации совместной работы и версионирования

Шаблон "Слои" — классический шаблон

↑ высокоуровневая организация

Советы по поиску пакетов АНАЛИЗА

- Искать группы связных классов
- иерархии наследования

Где искать: модели анализа
модели прецедентов

Главный принцип ПРОСТОТА

Важно избегать циклических зависимостей!

Потому что зависимости могут оказаться "бесконечными"

=> часть кода выносим в новый пакет — избегаем циклов

Зависимости пакетов UML

«use» — (по умолчанию) эл-т из клиента использует эл-т из поставщика

«import» — открытые эл-ты поставщика добавляются как открытые в пр-во имён клиента

«access» — открытые эл-ты поставщика доб-ся как закрытые в пр-во имён клиента

«trace» — трассируемость из одной модели в другую
(модель анализа <---- модель проектирования)

«merge» — только для мета-моделей, не используем

+ Возможность объединения пакетов

→ След. шаг

② Механизм анализа

Механизмы в УП:

(примеры)

1) Аналитические механизмы (концептуальные)
2) Проектные механизмы (конкретика)
3) Механизмы реализации (исходный код)

Механизм — "набор кода" — определяет функциональность, но не появляется благодаря конкретному требованию

1) Аналит. мех-мы — "системные понятия"

- Persistence
- Communication
- Message routing
- Distribution
- Transaction management
- Process control and synchronization
- Information exchange, format conversion
- Security
- Error detection/handling/reporting
- Redundancy
- Legacy interface

* Можно указывать подробности реализации механизмов
* Бу текстовые примечания

③ Ключевые абстракции

– сущности модели предметной области

④ Создание реализации прецедентов

* Осмысление прецедентов НЕ ВСЕ сразу
* На 1 итерацию – не более 10-15% прец. системы
* Начинать с наиболее важных
* Создают реализацию проектировщики

Задачи работы «Анализ прецедентов»

– Выявить аналитические классы, необходимые для реализации прец-ов. VOPC

– Присвоить классам анализа стереотипы «boundary», «control», «entity»

– Описать взаимодействие классов

– По аналитическим классам распределяются обязанности

Источники аналитических классов

- Спецификации прецедентов
- МПО
- Словари
- Запросы заказчика
- Описания нефункц. требований
- Документ-концепция/ТЗ
- Другие док-ты проекта

Аналитические классы

* содержат
  - ключевые атр-ты
  - обязанности

Инкапсулируют взаимодействие системы с внешним миром

- Классы пользовательских интерфейсов
- Классы системных

Инкапсулируют данные и алгоритмы предметной области

* Содержат „пользу"
* Для выявления используем МПО

интерфейсов
- Класс аналогичных интерфейс

Инкапсулирует выполнение
сравнительно
прецедентов
к не выполняет содержа-
тельных действий,
и „поручает" выполнение

Хороший аналитический класс

- Короткое осмысленное имя
- Четкая абстракция, один эл-т ПО
- Немного (одна) чётко определенных обязанностей
- Высокая внутр сплоченность
- Низкая внешняя связанность

Рекомендации по выявлению А. К.

- Чем проще, тем лучше
- Не более 3-5 обязанностей на класс
- Нет суффиксов классов
- Соблюдается баланс пассивности
- Избегайте функционалов
- Избегайте классов-повелителей
- Не более 3-4 уровней наследования

Class Responsibility Collaborators

1.1. Мозговой штурм – идеи о классах

1.2. Все заинтересованные стороны, относящиеся к области их компетенции

1.3. Класс команды разработки формулируют и записывают обязанности классов

1.4 Выявляют класс-сообщники

2. Эксперты команды анализируют результаты мозгового штурма

Имя класса – Class

Обязанности	Сообщники
	CRC-анализ

+ Другая техника: поиск существ в спец (не очень.)

Стереотипы R&P и MVC

- Model ⟷ entity класс

- View ⟷ boundary class
- Controller ⟷ control класс

Реализация прецедента

— чтобы сформулировать, каким образом обеспечивается услуга системы

VOPC — содержат класс анализа

* Задача: распределить обязанности анализ. классов
  — by диаг взаимодействия
  ↑
  диаг посл-ти
  ↑
  диаг коммуникации

Диаграмма взаимодействия UML

Пример:

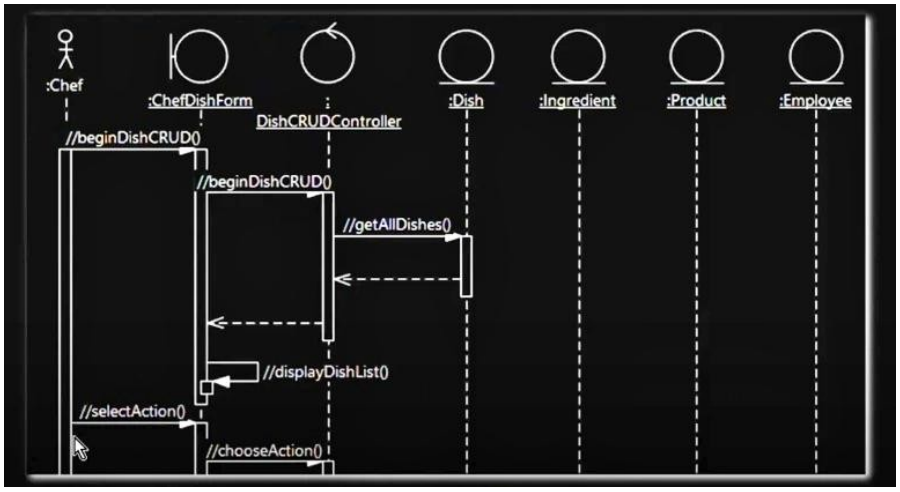
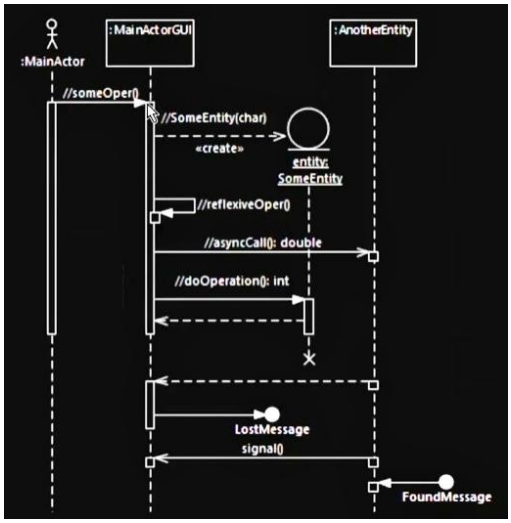


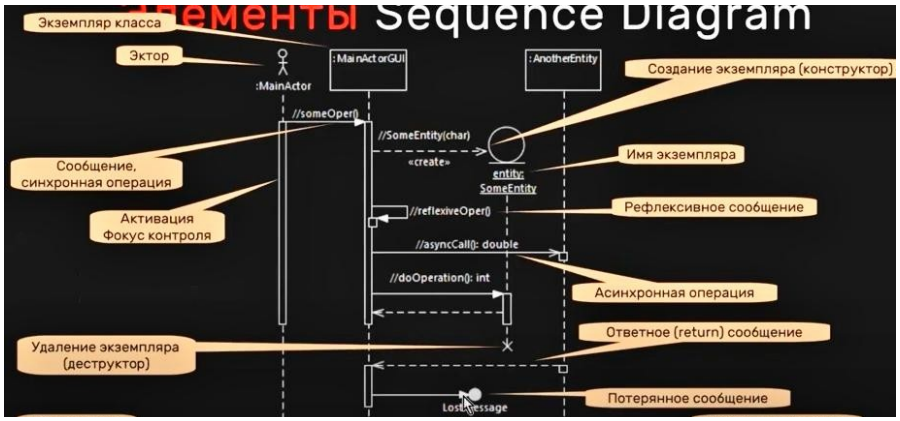
Диаграмма послед-ти:

- life line, · name — экземпляр объекта

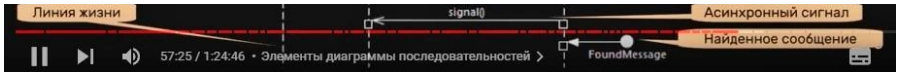
Элементы Sequence Diagram



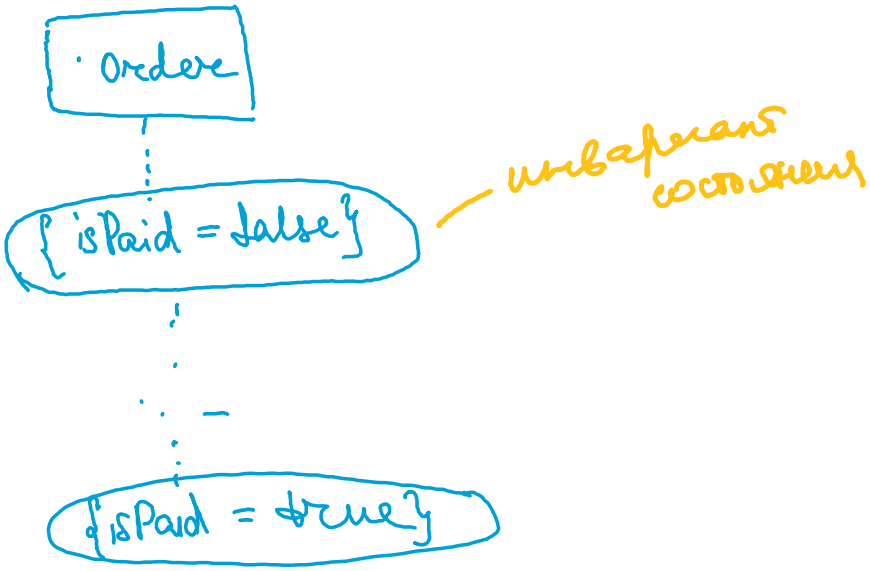
1) mainActor обращается с запросом к GUI
2) GUI создает сущность



Пояснения:



Инварианты состояния



Операторы ветвления и цикла

* Условия должны быть взаимоисключающими

Комбинирование фрагментов

ТАБЛИЦА

	Оператор		Семантика
	Option	1	если {условие истина} тогда {выполнить}
	Alternatives	2	выполнить тот блок, для которого {условие истина}
	Loop	3	loop min,max [условие] – повторить min раз, потом повторять пока {условие истина} max-min раз
	Break	4	если условие истина, выполнить блок, а не всё остальное
	Reference	5	ссылка на другой объект взаимодействия
	Parallel	6	все блоки выполнять параллельно
	Critical	7	выполнить блок без прерываний
	Weak sequencing	8	выполнить блоки параллельно при условии, что последовательность поступления сообщений на одну линию жизни от всех блоков такая же, как и последовательность блоков
	strict sequencing	9	блоки выполняются строго последовательно
	Negative	10	ошибочное поведение (пример «как не может быть»)
	Ignore	11	сообщения фрагмента игнорируются при взаимодействии
	Consider	12	сообщения фрагмента рассматриваются для использования

Диаграммы коммуникации.

Элементы:

* По набору информации не отличаются
* Используются реже

Четыре типа диаграмм взаимодействия

1. Диаграммы последовательности
   - моделирование посл-ти событий
2. Диаграммы коммуникации
   - структурно-поведенческие взаимосвязи
3. Диаграммы обзора взаимодействия
   - Узлы состоятся на взаимодействия
4. Временные диаграммы
   - Удобны для анализа продолжительности

→ Применяются в задачах проектирования

Пример „Колледж"

* Основной поток и регистрация
↓
с прошлой лекции — спецификацию