Принципы и паттерны объектно-ориентированного программирования

Паттерны, банда четырех

Нотация OMT, UML

Классификация паттернов

Ключевые принципы для повторного использования

(интерфейсы, композиция)

Паттерны проектирования

- Паттерны проектирования – описание взаимодействия объектов и классов, адаптированных для решения общей задачи проектирования в конкретном контексте
- Обобщенное описание решения задачи, которое можно использовать в различных ситуациях
- Паттернами не являются: конкретные классы, алгоритмы
- Под *классическими паттернами* проектирования понимаются повторяющиеся элементы дизайна приложений для объектно-ориентированных языков программирования со статической типизацией (C++, C#, Java, Object Pascal и др.)

Для чего применяются паттерны

- Для создания «хорошего дизайна»
  - Инкапсуляция
  - Повторное использование
  - Снижение зависимостей
  - И т.д.
- **Проверенные решения.** «Вы тратите меньше времени, используя готовые решения, вместо повторного изобретения велосипеда».
- **Стандартизация кода.** «Вы делаете меньше просчётов при проектировании, используя типовые унифицированные решения, так как все скрытые проблемы в них уже давно найдены».
- **Общий программистский словарь.** «Вы произносите название паттерна, вместо того, чтобы час объяснять другим программистам, какой крутой дизайн вы придумали и какие классы для этого нужны».

Канонические паттерны (GoF-паттерты)

- GoF – Gang of Four («банда четырех») – Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес
- Книга «Design Patterns: Elements of Reusable Object-Oriented Software» («Приемы объектно-ориентированного проектирования. Паттерны проектирования»)
  - Вышла в 1995 году
  - Выделены 23 паттерна проектирования (получили название Go. F-паттернов или канонических паттернов)
  - Примеры: C++, SmallTalk

Классификация GoF-паттернов

- **Порождающие паттерны**
  - Отвечают за создание объектов, позволяя системе оставаться независимой как от самого процесса порождения, так и от типов порождаемых объектов
- **Структурные паттерны**
  - Организуют структуру классов (на этапе разработки) или объектов (на этапе выполнения программы)
- **Паттерны поведения**
  - Характеризуют, как классы и объекты взаимодействуют между собой

Пространство паттернов GoF

Цель / Уровень	Порождающие паттерны	Структурные паттерны	Паттерны поведения
Класс	Фабричный метод	Адаптер (класса)	Интерпретатор
			Шаблонный метод
Объект	Абстрактная фабрика	Адаптер (объекта)	Итератор
	Одиночка	Декоратор	Команда
	Прототип	Заместитель	Наблюдатель
	Строитель	Компоновщик	Посетитель
		Мост	Посредник
		Приспособленец	Состояние
		Фасад	Стратегия
			Хранитель
			Цепочка обязанностей

Графическая нотация OMT -> UML

- OMT (Object Modeling Technique) – один из методов ООАП и одновременно одна из общепризнанных систем графических обозначений Д. Румбаха (James Rumbaugh)
- OMT привело к созданию UML – унифицированного языка графического описания программных систем, организационных систем.
- UML использует большое разнообразие диаграмм для разных сценариев проектирования и описания.

Диаграммы в UML

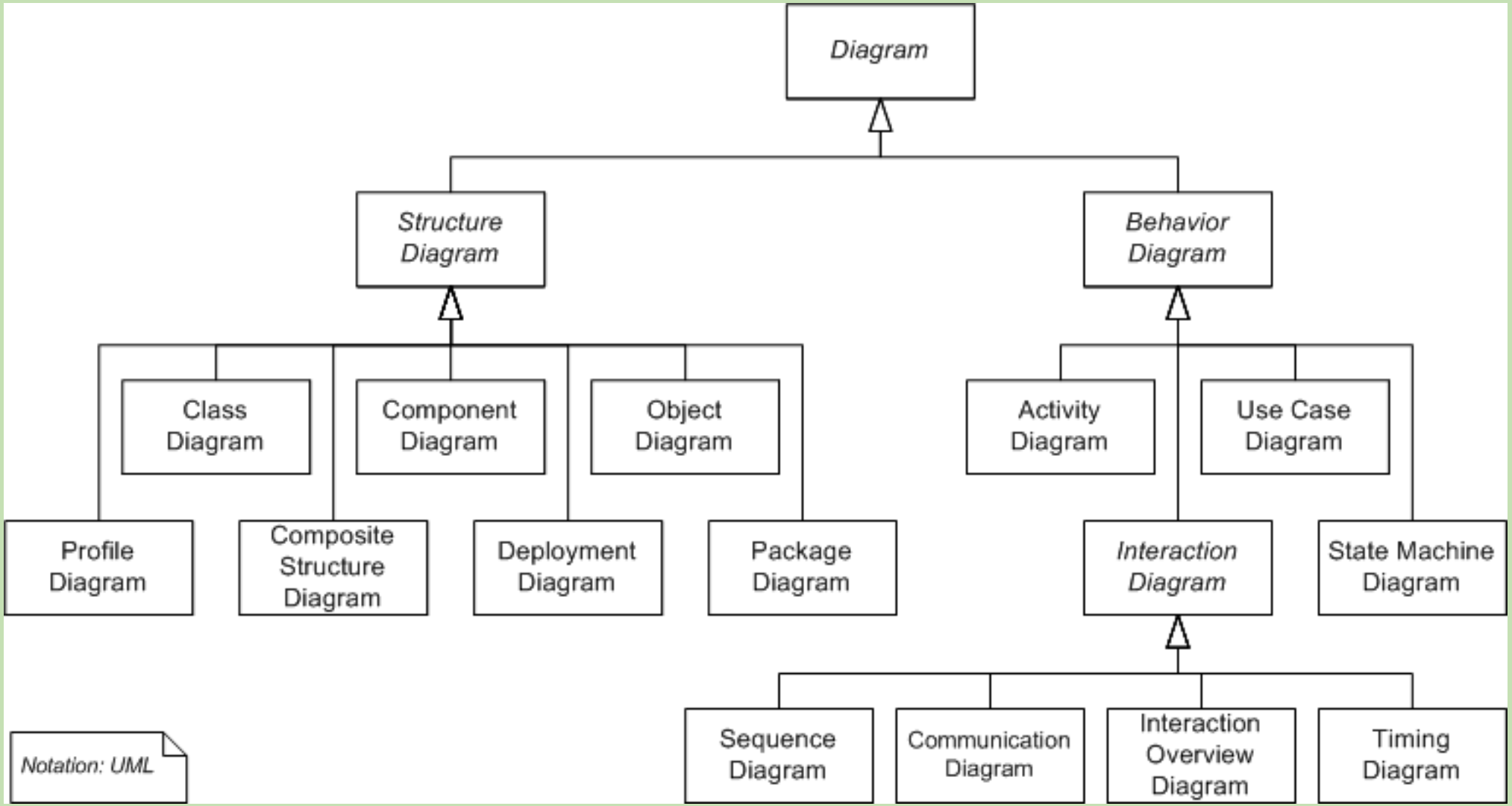
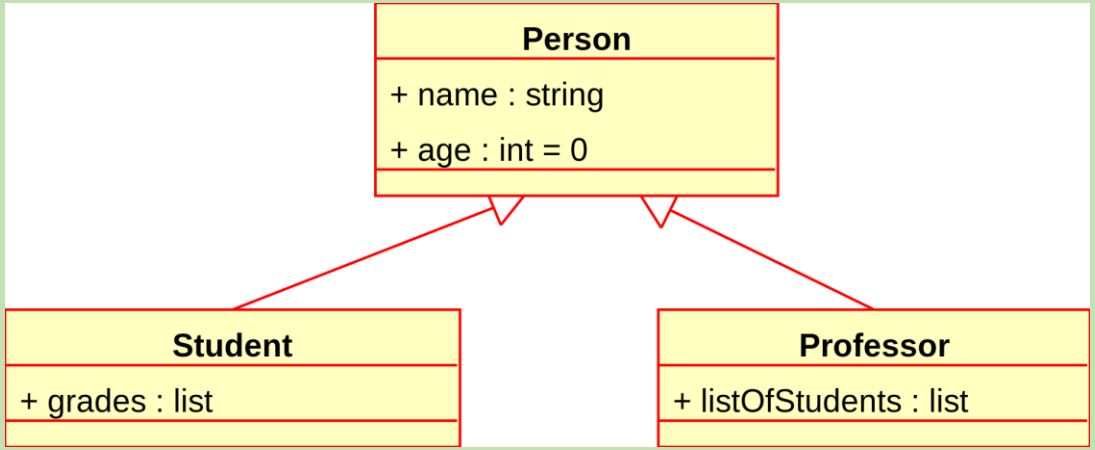


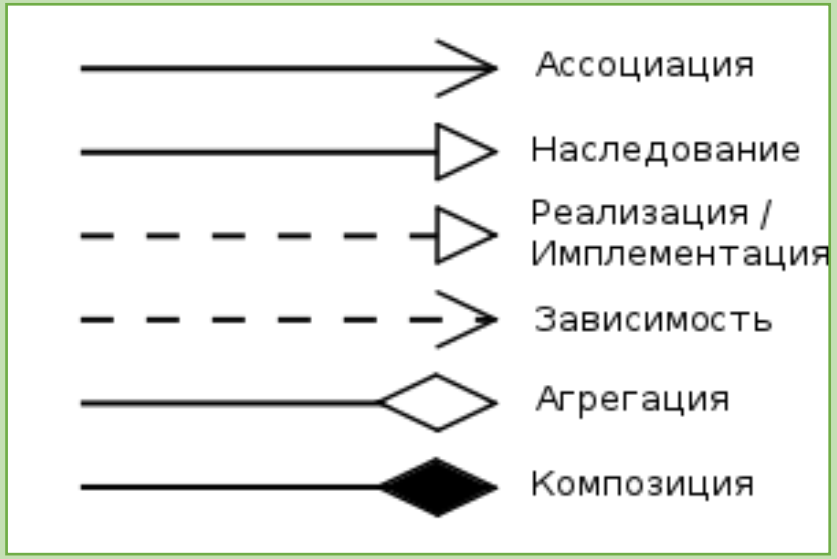
Диаграмма классов

- Графическое отображение статической структуры декларативных элементов системы
- Уровни отображения:
  - Аналитический (концептуальный)
  - Уровень проектирования
  - Уровень реализации



Виды отношений между классами

- UML позволяет отображать разные типы связей между классами
- Наследование в ООП в UML отображается с помощью двух стрелок (наследование, реализация)
- Отношения «часть-целое»:
  - Ассоциация (человек-школа).
  - Агрегация (комната-стул);
  - Композиция (комната-квартира);
- Зависимость: обозначает такое отношение между классами, что изменение спецификации класса-поставщика может повлиять на работу зависимого класса, но не наоборот.

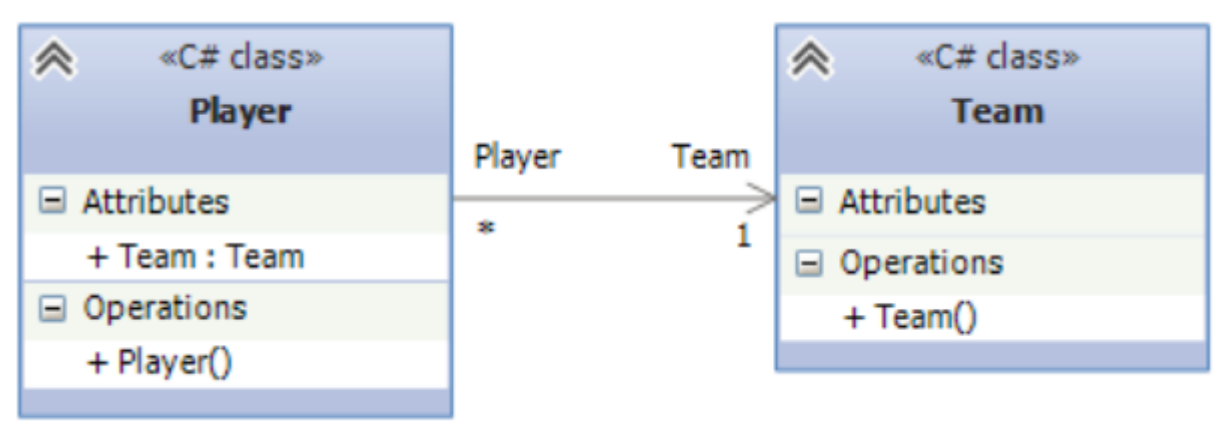


Ассоциация

- Ассоциация - это отношение, при котором объекты одного типа каким-то образом связаны с объектами другого типа (содержит или используют).
- Моделирует отношение «имеет», «состоит из»:

  автомобиль имеет двигатель

  футбольный клуб состоит из игроков



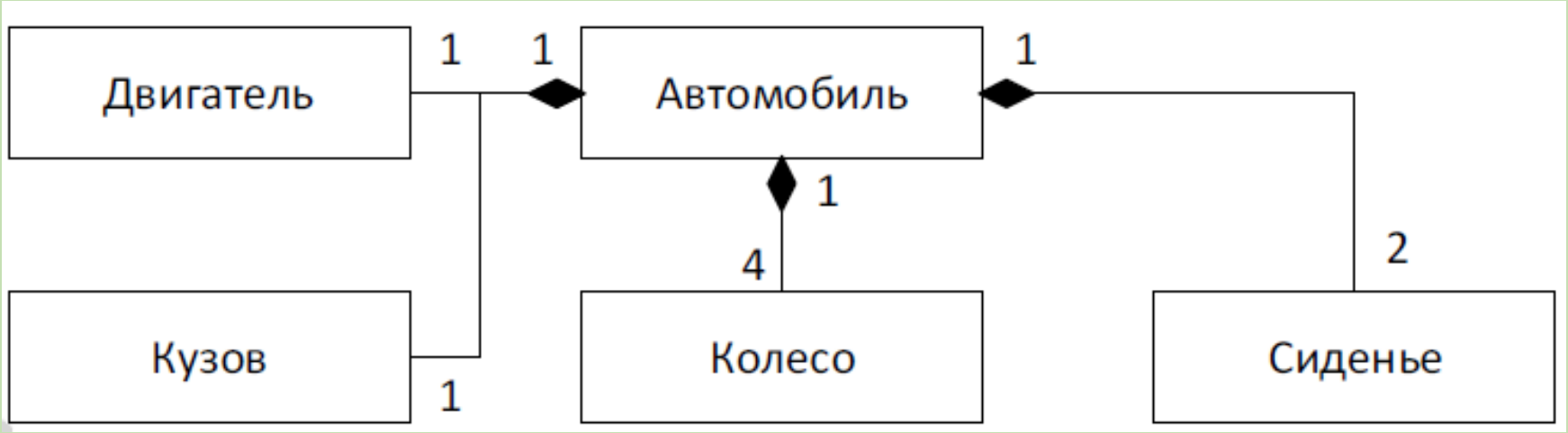
- Связь между сущностями может быть достаточно жесткой или слабой (объекты существуют сами по себе)
- Частные случаи ассоциации, которая реализует отношение “has ”: композиция и агрегация

Агрегация

- Агрегация - это тип отношений, когда один объект является частью другого. Агрегация образует слабую связь между объектами. Все зависимые классы инициализируются вне основного объекта.

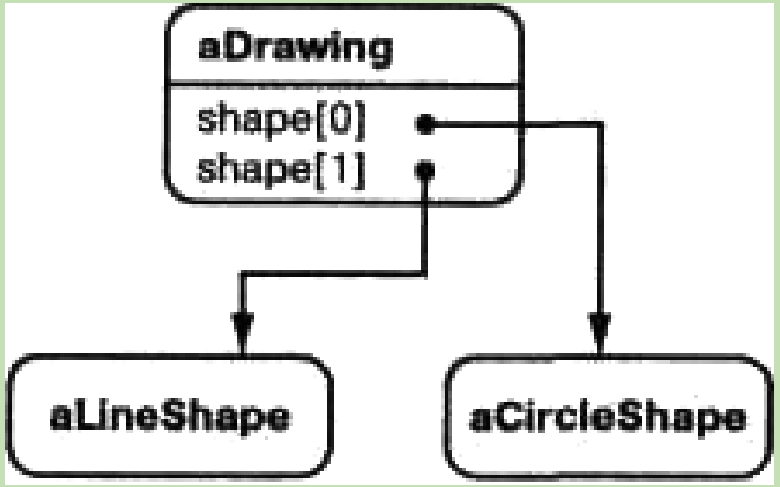
Композиция

- Форма агрегации, при которой делается акцент на том, что часть не может существовать в отрыве от целого. Уничтожается целое – уничтожается и часть.



Диаграммы объектов, OMT

- В диаграммах объектов приводятся только экземпляры в какой-то определенный момент времени
- Для обозначения объектов используются прямоугольники со скругленными углами
- Объекты именуются «aSomething», где Something – класс объекта
- Стрелки ведут к объектам, на которые ссылается данный



Диаграммы последовательности (взаимодействия)

- Время на диаграммах взаимодействий откладывается сверху вниз
- Сплошная вертикальная черта означает время жизни объекта, до момента создания объекта вертикальная линия идет пунктиром
- Запросы, посылаемые другим объектам (вызовы методов), обозначаются горизонтальной стрелкой, указывающей на объект получатель; имя запроса (метода) показывается над стрелкой
- Виды стрелок:
  - синхронное сообщение (закрытая, сплошная);
  - ответное сообщение (пунктирная линия);
  - асинхронное сообщение (открытая стрелка)