



Programming
Languages

Лабораторная работа #27

**Объектно-Ориентированное Программирование.
Абстрагирование/абстракция.
Инициализация объектов
в языке Python**



LEARN. GROW. SUCCEED.

© 2019-2020. Department: <Software of Information Systems and Technologies>
Faculty of Information Technology and Robotics
Belarusian National Technical University
by Viktor Ivanchenko / ivanvikvik@bntu.by / Minsk

ЛАБОРАТОРНАЯ РАБОТА #27

Абстрагирование/абстракция в ООП. Инициализация объектов в языке Python

Цель работы

Научиться грамотно анализировать предметную область и с помощью абстракции (абстрагирования) выделять существенные детали, на базе которых в дальнейшем проектируются классы и создаются объекты будущей программной системы согласно методологии ООП.

Общее задание

Необходимо произвести небольшую модернизацию предметной области и программы, созданных в предыдущей лабораторной работе, следующим образом:

- расширить набор атрибутов основной сущности предметной области (их может быть и больше);
- перегрузить соответствующие конструкторы и добавить деструктор в соответствующие классы;
- проанализировать доступные способы и средства первоначальной инициализации состояния объектов в языке Python;
- на уровне всей программы добавить глобальный механизм отслеживания количества создаваемых объектов предметной области, которыми манипулирует система при выполнении основных расчётов программной системы.

Для тех, кто ещё не определился с нормальной предметной областью, есть возможность выбрать её из списка ниже.



Требования к выполнению задания

- 1) Необходимо модернизировать спроектированную в предыдущей лабораторной работе UML-диаграмму взаимодействия классов и объектов программной системы исходя из текущих дополнений. На базе данной изменённой UML-диаграммы реализовать рабочее приложение с использованием архитектурного шаблона проектирования **MVC**.
- 2) Каждый класс разрабатываемого приложения должен иметь адекватное осмысленное имя (обычно это *имя существительное*). Имена полей и методов должны нести также логический смысл (имя метода, который что-то вычисляет, обычно называют *глаголом*, а поле – именем существительным). Имя класса пишется с большой (заглавной) буквы, а имена методов и переменных – с маленькой (строчной).
- 3) Соответствующие классы должны группироваться по модулям, которые затем подключаются там, где происходит создание объектов классов и их использование.
- 4) При проектировании классов необходимо придерживаться принципа единственной ответственности (**Single Responsibility Principle**), т.е. классы должны проектироваться и реализовываться таким образом, чтобы они были слабо завязаны с другими классами при своей работе – они должны быть самодостаточными.
- 5) При выполнении заданий необходимо по максимуму пытаться разрабатывать универсальный, масштабируемый, легко поддерживаемый и читаемый код.
- 6) В соответствующих компонентах (классах, функциях) бизнес-логики необходимо предусмотреть «защиту от дурака».
- 7) Рекомендуется избегать использования глобальных переменных.
- 8) Там, где это необходимо, необходимо обеспечить грамотную обработку исключительных ситуаций, которые могут произойти при выполнении разработанной программы.
- 9) Все действия, связанные с демонстрацией работы приложения, должны быть размещены в главном модуле программы в функции **main**. При проверки работоспособности приложения необходимо проверить все тестовые случаи.



- 10) Программы должны обязательно быть снабжены комментариями на английском языке, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчика, номер группы и дату разработки. Исходный текст программного кода и демонстрационной программы рекомендуется также снабжать поясняющими краткими комментариями.
- 11) Программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом для взаимодействия с пользователем. Интерфейс программы должен быть на английском языке.
- 12) При разработке программы придерживайтесь соглашений по написанию кода на Python (***Python Code Convention***).

Best of LUCK with it, and remember to HAVE FUN while you're learning :)

Victor Ivanchenko



Примеры предметных областей

1. **Цветочница или магазин цветов (*Flower Shop*)**. В магазине цветов можно собрать букет из соответствующих цветов. Необходимо определить вес букета, его стоимость и самый дорогой цветок.
2. **Новогодняя ёлка (*Christmas tree*)**. Есть новогодняя ёлка, которую можно украсить соответствующими новогодними игрушками. Необходимо подсчитать вес всей ёлки вместе с игрушками, её общую стоимость, а также найти самую дорогую игрушку.
3. **Кредиты (*Credits*)**. В банке можно сформировать набор предложений клиенту по соответствующим целевым кредитам. Необходимо подсчитать общую сумму по всем кредитам клиента и найти максимальную ставку по кредиту и самый дорогой кредит клиента.
4. **Жилищно-Коммунальное Хозяйство, ЖКХ (*Housing and Communal Services*)**. В ЖКХ предлагают набор соответствующих услуг по обслуживанию и эксплуатации жилищного хозяйства клиента. Необходимо подсчитать общую стоимость услуг, которые были оказаны клиенту ЖКХ и найти самую дорогую (недорогую) услугу.
5. **Турагентство и Туристические путевки (*Tourist trips*)**. В турфирме можно сформировать набор предложений клиенту по выбору туристической путевки различного типа. Необходимо подсчитать общую сумму, которую клиент должен заплатить за выбранные им путёвки и найти самую выгодную по стоимости путёвки, исходя из расчёта стоимости за один день.
6. **Шеф-повар (*Chef*)**. Необходимо приготовить овощной салат и определить вес салата и его калорийность, а также найти самый калорийных овощ в салате.
7. **Шеф-повар (*Chef*)**. Необходимо приготовить фруктовый салат и определить вес салата и его калорийность, а также найти самый калорийных фрукт в салате.
8. **Налоги (*Taxes*)**. Определить множество и сумму налоговых выплат физического лица за год с учетом доходов с основного и дополнительного мест работы, авторских вознаграждений, продажи имущества, получения в подарок денежных сумм и имущества, переводов из-за границы, льгот на детей и материальную помощь.



9. **Шеф-повар (Chef)**. Необходимо приготовить основное блюдо (или десерт) определить его вес и калорийность, а также найти самый калорийных ингредиент в блюде (десерте).
10. **Игровая комната (Game room)**. Есть игровая комната с соответствующими игрушками. Необходимо провести подсчёт стоимости всех игрушек, их общий вес и найти самую дорогую игрушку.
11. **Таксопарк (Taxi Station)**. Есть таксопарк, который состоит из соответствующего автотранспорта. Необходимо подсчитать стоимость автопарка и найти самый дорогой автотранспорт по тарифам поездки.
12. **Авиакомпания (Airline)**. Есть авиакомпания, которая состоит из соответствующих самолётов. Необходимо подсчитать общее количество мест для пассажиров и грузоподъёмность всех самолётов авиакомпании, а также найти самый вместительный самолёт авиакомпании.
13. **Камни (Stones)**. В ювелирном магазине можно отобрать соответствующие драгоценные камни для ожерелья. Необходимо подсчитать общий вес (в каратах) и стоимость изделия. Определить самый дорогой камень в ожерелье.
14. **Звукозапись (Sound Recording)**. В звукозаписывающей студии можно записать на диск сборку музыки. Необходимо подсчитать продолжительность всего диска и найти самую короткую (длинную) песню.
15. **Компьютерный герой (Game Hero)**. Определить сильные стороны героя (его способности: ум, сила, ловкость и т.д.) на базе собранных им артефактов. Подсчитать общую стоимость артефактов, а также величину его соответствующих способностей.
16. **Новогодний подарок (New Year gift)**. Есть новогодний сладкий подарок, который состоит из конфет и прочих сладостей. Необходимо подсчитать общий вес и стоимость подарка, а также самую дорогую (недорогую) сладость.
17. **Железнодорожный транспорт (Railway Transport)**. Имеются поезда, которые состоят из соответствующих вагонов. Необходимо подсчитать общую длину конкретного поезда, а также найти самый длинный (короткий) вагон.
18. **Автосалон (Car Center)**. Есть автосалон, который состоит из соответствующих машин. Необходимо подсчитать общую стоимость машин автосалона и найти самую дорогую (недорогую) машину.



19. **Страховое агентство (*Insurance Company*)**. Есть страховая фирма, которая предлагает страховые услуги и обязательства своим клиентам. Необходимо подсчитать общую стоимость страховых услуг, оказываемых конкретному клиенту, а также найти самую дорогую (недорогую) услугу.
20. **Грузоперевозки (*Cargo Transportation*)**. Есть транспортная компания, которая занимается грузовыми перевозками. Необходимо подсчитать максимальное количество грузов, которое за один раз может осуществить компания, исходя из имеющего собственного грузового транспорта, а также найти транспорт, который перевозит максимальное (минимальное) количество груза.
21. **Пассажирские перевозки (*Passenger Operations*)**. Есть транспортная компания, которая занимается пассажирскими перевозками. Необходимо подсчитать максимальное количество пассажиров, которых за один раз может перевезти компания, исходя из имеющего собственного пассажирского транспорта, а также найти транспорт, который перевозит максимальное (минимальное) количество пассажиров.
22. **Фургон кофе (*Coffee Car*)**. Есть фургон определенного объема, в который можно загрузить на определенную сумму соответствующие упаковки кофе. Необходимо определить, сколько упаковок может влезть в фургон заданного объема и на какую общую сумму.
23. **Рыцарь (*Knight*)**. Есть рыцаря, который экипирован соответствующей амуницией. Необходимо определить общую стоимость всей амуниции, а также найти самую дорогую (недорогую) амуницию рыцаря.
24. **Мобильная связь (*Mobile Communication*)**. Компания-оператор мобильной связи предлагает своим клиентам различные тарифные планы использования мобильной связи. Необходимо определить общую сумму дохода компании в месяц, которую она получает от клиентов за использования соответствующих тарифных планов, а также найти клиента, который заплатил больше (меньше) других.
25. **Футбольный Менеджер (*Football Manager System*)**. Есть футбольные клубы, у которых есть соответствующие характеристики, влияющие на исход встречи с противником. Необходимо определить шансы футбольной команды на победу против команды-соперника, а также общие шансы на победу в кубке и чемпионате соответствующей страны.



26. **Вклады или кредиты (*Deposits or Credits*)**. Банки предлагают своим клиентам соответствующие вклады или кредиты с различными процентами. Необходимо определить общую сумму дохода соответствующего банка в месяц, которую он получает согласно соответствующей марже. Найти самые дорогие (недорогие) кредиты с учётом всех платежей или самые выгодные (невыгодные) вклады с учётом процентной ставки и доходности по ним.
27. **Зоопарк (*Zoo*)**. Есть зоопарк животных, которые **ОЧЕНЬ** любят кушать. Необходимо определить общий суточный запас продуктов, необходимых зоопарку, чтобы прокормить всех своих животных. Также необходимо найти самых (мелее) прожорливых животных зоопарка.
28. **Спортивная рыбалка (*Sports Fishing*)**. В соревнованиях по спортивной рыбалке собираются команды из нескольких рыбаков. В процессе соревнований каждая из команд за определённое время должна наловить максимальное количество рыбы. Необходимо определить общий улов всех команд, а также найти команду победителя (команда, которая имеет наибольший улов рыбы) и аутсайдера (команда, которая имеет самые скромные результаты по улову).
29. **Компания по разработки программного обеспечения, ИТ-компания (*Software Company*)**. Есть ИТ-компании, которые состоят из соответствующих сотрудников. Каждый сотрудник за свою работу получает соответствующую зарплату. Необходимо найти общий фонд заработной платы компании на месяц, а также определить сотрудников с максимальной (минимальной) заработной платой.
30. **Домашние электроприборы (*House Equipments*)**. Подсчитать потребляемую мощность всех бытовых приборов или только тех, которые включенный в данный момент. Найти также бытовой прибор, который наиболее (наименее) энергозатратный.
31. **Пчелиная пасека (*Bee Apiary*)**. На пасеке есть несколько домиков с ульями пчёл. Каждая пчела собирает за день определённое количество мёда и приносит его в свой улей. Необходимо подсчитать общее количество мёда в день, которая даёт вся пасека, а также определить улей, которые даёт больше (меньше) всего мёда.



32. **Поликлиника (*Clinic/Hospital*)**. В городе есть несколько поликлиник (больниц), куда обращаются пациенты, у которых есть проблемы со здоровьем. Необходимо подсчитать общее количество обращений во все поликлиники (больницы) города, а также найти самую (менее) загруженную по посещению поликлинику (больницу).
33. **Библиотека (*Library*)**. Библиотека имеет собственный книжный фонд и предоставляет своим читателям книги (и другую литературу), которые можно почитать как в самой библиотеки, так и взять с собой на определённый период. Необходимо подсчитать, сколько сейчас книг на руках у читателей, а также определить самую (менее) популярную книгу у читателей.
34. **Тюрьма (*Jail/Prison*)**. Есть тюрьма, в которой содержатся заключённые, осуждённые по разным статьям и на различный срок. Необходимо подсчитать суммарное количество дней (или месяцев, или лет, ...) всех заключённых, а также найти самую (менее) популярную уголовную статью, за которую отбывают наказание осуждённые, или найти самый большой (короткий) срок за заключённого.
35. **Морской порт (*Sea Port*)**. Есть морской порт с пирсами (причалами), на которых каждый день происходит выгрузка и загрузка соответствующих грузов. Необходимо подсчитать общий грузопоток, который проходит через данный порт за сутки (или любой другой период), а также определить максимально (минимально) загруженный пирс (причал).
36. **Видео/Компьютерная игра (*Video/Computer Game*)**. Есть крутая игра, направленная на зарабатывание игроками в процессе игры определённого количества «качества» – это может быть соответствующие баллы, время, уровни, виртуальные деньги, алмазы, брильянты, золото, другие виртуальные ресурсы и т.д. Необходимо определить максимальное количество «качества», заработанное всеми игроками в данной видео игре, а также выявить победителя (– игрок с максимальным количеством «качества») и аутсайдера (– игрок с минимальным количеством «качества») на данный момент.
37. **Продуктовый магазин или любой другой магазин (*Grocery Store*)**. Есть магазин, в который каждый день приходят покупатели за продуктами. Необходимо подсчитать среднюю выручку магазина за сутки (или за месяц, или год и т.д.), а также найти покупателя, который оставил в магазине само больше (меньше) денег за отчётный период.



38. **Ресторан (*Restaurant*)**. Есть ресторан соответствующей кухни, который своим посетителям предоставляет разнообразное меню блюд. Необходимо подсчитать общую сумму заказа соответствующего столика или общее количества денег, который ресторан получил за отчётный период (день, месяц и т.д.). Также необходимо найти чек (столлик) с максимальной (минимальной) суммой заказа.
39. **Автостоянка (*Parking*)**. В городе есть автостоянка (или несколько автостоянок), стоимость машиномест на которой тарифицируется по часам. Необходимо подсчитать общую сумму, которую получает автостоянка за сутки (или любой другой отчётный период), а также автовладельца, который суммарно заплатил больше (меньше) всего денег за парковку.
40. **Суд или судебное делопроизводство (*Court*)**. В городе есть суд, который каждый день рассматривает уголовные дела или другие правонарушения, касающиеся нарушения закона. Необходимо подсчитать: общее количество дел, которые были рассмотрены судом за отчётный период; сколько из общих дел были оправдательными, а сколько – с доказанной виной и повлёкшим к заключению под стражу виновных. Можно дополнительно подсчитать общее число лиц, которые были осуждены на соответствующие сроки и сейчас находятся в местах отбывания наказания, а также найти осуждённых, которым судья установил максимальный (минимальный) срок.



Что нужно запомнить (краткие тезисы)

1. Все экземпляры классов создаются в динамической памяти – в хипе (heap).
2. **Каждый класс** в ООП может иметь специальные методы, которые автоматически вызываются при создании и(или) уничтожении экземпляров класса:
 - **конструктор (*constructor*)** – служит для первоначальной инициализации состояния объекта и выполняется сразу же после создания объекта в памяти;
 - **деструктор (*destructor*)** – служит для освобождения всех ресурсов, которые были выделены для объекта, и выполняется перед полным удалением объекта из памяти сборщиком мусора (*garbage collector*, GC);
3. В упрощённом виде, класс – это кусок кода, у которого есть имя. Чтобы воспользоваться данным кодом, нужно создать объект (**экземпляр класса**).
4. **Инстанцирование** – процесс создания объекта (экземпляра класса).
5. Обычно в классе можно описать (перегрузить) несколько конструкторов. При создании экземпляра объекта необходимо явно указать нужный конструктор, с помощью которого будем осуществляться инициализация первоначального состояния объекта.
6. Как правило, деструктор в классе только один.
7. Удаление ненужных объектов из динамической памяти в Python осуществляется автоматически специальным движком – ***Garbage Collector*** (GC).
8. **Основные столпы (парадигмы) ООП** (порядок перечисления очень важен!):
 - 1) абстракция;
 - 2) инкапсуляция;
 - 3) наследование;
 - 4) полиморфизм;
 - 5) посылка сообщений (вызов соответствующих методов, свойств и т.д.);
 - 6) повторное использование кода.
9. **Абстрагирование в ООП** – это способ выделить набор значимых характеристик объекта, исключая из рассмотрения незначимые. Абстракция – это набор всех таких характеристик.
10. **Абстракция в ООП на уровне объекта** – это придание объекту характеристик, которые отличают его от всех других объектов, чётко определяя его концептуальные границы.



Графическое представление элементов UML-диаграммы классов

UML – унифицированный язык моделирования (***Unified Modeling Language***) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для **визуализации, спецификации, конструирования** и **документирования** программных систем. Язык UML применяется не только для проектирования, но и с целью документирования, а также эскизирования проекта.

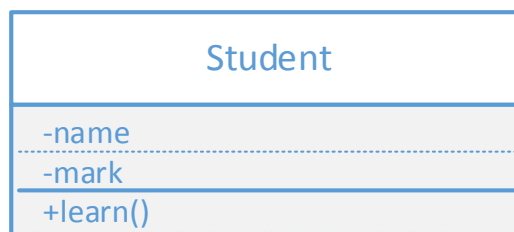
Словарь UML включает три вида строительных блоков: **диаграммы, сущности** и **связи**. **Сущности** – это абстракции, которые являются основными элементами модели, **связи** соединяют их между собой, а **диаграммы** группируют представляющие интерес наборы сущностей.

Диаграмма – это графическое представление набора элементов, чаще всего изображенного в виде связанного графа вершин (сущностей) и путей (связей). Язык UML включает **13** видов диаграмм, среди которых на первом (центральном) месте в списке – диаграмма классов.

UML-диаграмма классов (*Static Structure Diagram*) – диаграмма статического представления системы, демонстрирующая классы (и другие сущности) системы, их атрибуты, методы и взаимосвязи между ними.

Диаграммы классов оперируют тремя видами сущностей UML: структурные, поведенческие и аннотирующие.

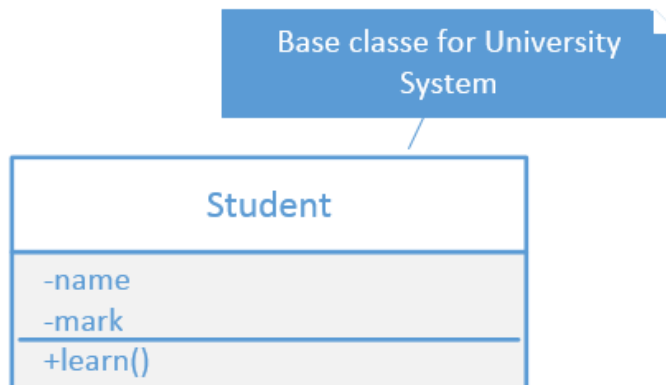
Структурные сущности – это «имена существительные» в модели UML. В основном, статические части модели, представляющие либо концептуальные, либо физические элементы. Основным видом структурной сущности в диаграммах классов является класс. Пример класса Студент (*Student*) с полями и методами:



Поведенческие сущности – динамические части моделей UML. Это «глаголы» моделей, представляющие поведение модели во времени и пространстве. Основной из них является взаимодействие – поведение, которое заключается в обмене сообщениями между наборами объектов или ролей в определенном контексте для достижения некоторой цели. Сообщение изображается в виде линии со стрелкой:



Аннотирующие сущности – это поясняющие части UML-моделей, иными словами, комментарии, которые можно применить для описания, выделения и пояснения любого элемента модели. Главная из аннотирующих сущностей – примечание. Это символ, служащий для описания ограничений и комментариев, относящихся к элементу либо набору элементов. Графически представлен прямоугольником с загнутым углом; внутри помещается текстовый или графический комментарий.



Графически класс изображается в виде прямоугольника, разделенного на 3 блока горизонтальными линиями: имя класса, атрибуты (свойства) класса и операции (методы) класса.

Для атрибутов и операций может быть указан один из нескольких типов видимости:

- + открытый, публичный (*public*)
- закрытый, приватный (*private*)
- # защищённый (*protected*)
- / производный (*derived*) (может быть совмещён с другими)
- ~ пакет (*package*)

Видимость для полей и методов указывается в виде левого символа в строке с именем соответствующего элемента.



Каждый класс должен обладать именем, отличающим его от других классов. **Имя** – это текстовая строка. Имя класса может состоять из любого числа букв, цифр и знаков препинания (за исключением двоеточия и точки) и может записываться в несколько строк. Каждое слово в имени класса традиционно пишут с заглавной буквы (верблюжья нотация), например Датчик (*Sensor*) или ДатчикТемпературы (*TemperatureSensor*).

Для **абстрактного класса** имя класса записывается **курсивом**.

Атрибут (свойство) – это именованное свойство класса, описывающее диапазон значений, которые может принимать экземпляр атрибута. Класс может иметь любое число атрибутов или не иметь ни одного. В последнем случае блок атрибутов оставляют пустым.

Атрибут представляет некоторое свойство моделируемой сущности, которым обладают все объекты данного класса. Имя атрибута, как и имя класса, может представлять собой текст. Можно уточнить спецификацию атрибута, указав его тип, кратность (если атрибут представляет собой массив некоторых значений) и начальное значение по умолчанию.

Статические атрибуты класса обозначаются **подчеркиванием**.

Операция (метод) – это реализация метода класса. Класс может иметь любое число операций либо не иметь ни одной. Часто вызов операции объекта изменяет его атрибуты.

Графически операции представлены в нижнем блоке описания класса.

Допускается указание только имен операций. Имя операции, как и имя класса, должно представлять собой текст. Каждое слово в имени операции пишется с заглавной буквы, за исключением первого, например move (переместить) или isEmpty (проверка на пустоту).

Можно специфицировать операцию, устанавливая ее сигнатуру, включающую имя, тип и значение по умолчанию всех параметров, а применительно к функциям – тип возвращаемого значения.

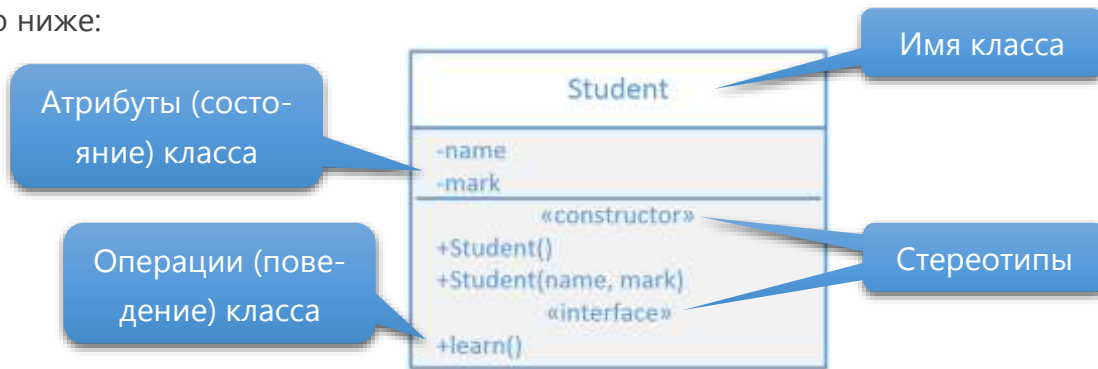
Абстрактные методы класса обозначаются **курсивным** шрифтом.

Статические методы класса обозначаются **подчеркиванием**.

Чтобы легче воспринимать длинные списки атрибутов и операций, желательно снабдить префиксом (именем стереотипа) каждую категорию в них. В данном случае **стереотип** – это слово, заключенное в угловые кавычки, которое указывает то, что за ним следует.



Общее описание класса с именем Student и другими атрибутами представлено ниже:



Существуют следующие типы связей в UML: **зависимость**, **ассоциация** (и её разновидности: **агрегация** и **композиция**), **наследование** (обобщение) и **реализация**. Эти связи представляют собой базовые строительные блоки для описания отношений в UML, используемые для разработки хорошо согласованных моделей.

Первая из них – **зависимость** – семантически представляет собой связь между двумя элементами модели, в которой *изменение одного элемента (независимого) может привести к изменению семантики другого элемента (зависимого)*. Графически представлена пунктирной линией, иногда со стрелкой, направленной к той сущности, от которой зависит еще одна; может быть снабжена меткой.



Зависимость – это связь *использования*, указывающая, что изменение спецификаций одной сущности может повлиять на другие сущности, использующие её.

Ассоциация – это структурная связь между элементами модели, которая описывает набор связей, существующих между объектами. Ассоциация показывает, что объекты одной сущности (класса) связаны с объектами другой сущности таким образом, что можно перемещаться от объектов одного класса к другому. Например, класс Человек (*Human*) и класс Школа (*School*) имеют ассоциацию, так как человек может учиться в школе. Ассоциации можно присвоить имя «учится в». В представлении однонаправленной ассоциации добавляется стрелка, указывающая на направление ассоциации.

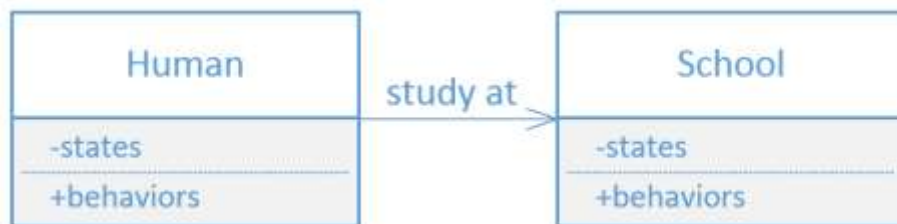
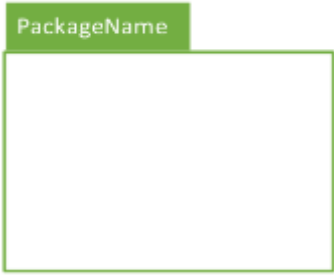
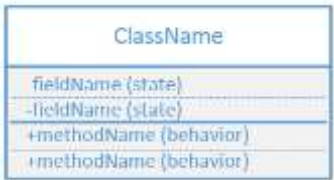
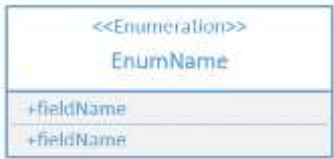



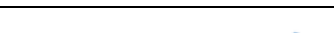
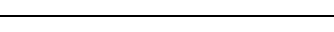


Таблица 1 – Наиболее часто используемые элементы UML-диаграммы

#	Shape (блок)	Description (описание)
1.		Элемент для описания пакета. Пакет логически выделяет группу классов, которые описаны в нём.
2.		Элемент для описания класса. Класс представлен в рамках, содержащих три компонента: имя класса, поля (атрибуты) класса и методы класса.
3.		Элемент для описания перечисления. Перечисление представляется аналогично классу с ключевым словом в самом вверху « <i>Enumeration</i> ».
4.		Элемент для описания интерфейса. Интерфейс представлен в рамках, содержащих два компонента: имя интерфейса с ключевым слово « <i>Interface</i> » и методы интерфейса.
5.		Аннотация (комментарий). Аннотация используется для размещения поясняющего (уточняющего) текста на диаграмме для соответствующих сущностей.
		Зависимость (<i>Dependency</i>)
6.		Ассоциация (<i>Association</i>)
7.		Агрегация (<i>Aggregation</i>)
8.		Композиция (<i>Composition</i>)
9.		Наследование (<i>Inheritance</i>)
10.		Реализация (<i>Implementation or Realization</i>)



Контрольные вопросы

1. Что такое инстанцирование?
2. Кто и как в языке Python создаёт экземпляр (объект) класса?
3. Как происходит выделения памяти под экземпляр класса? Приведите полную картину с учётом использования стека (stack) и хипа (heap)?
4. Как в терминах ООП называется метод, который вызывается сразу после создания объекта для инициализации его состояния?
5. Как описать конструктор в классе в языке Python?
6. Когда именно вызывается конструктор?
7. Можно ли перегружать конструкторы в языке Python? Если да, то зачем это нужно?
8. Кто занимается удалением объектов из памяти, если они уже не нужны системе?
9. Как в терминах ООП называется метод, который вызывается перед удалением объекта из памяти?
10. Как происходит удаление памяти объекта? Приведите полную картину с учётом использования стека (stack) и хипа (heap)?
11. Какие существуют основные столпы методологии ООП?
12. Что такое абстракция (абстрагирование) в ООП?
13. Каким образом можно сохранить данные на уровне системы не прибегая к глобальным переменным?
14. Чем отличаются атрибуты класса от атрибутов экземпляров класса?
15. Каким образом можно получить доступ к атрибутам класса? А как можно получить доступ к состоянию и поведению экземпляра класса?

