Programming Languages

Лабораторная работа #26

Основы Объектно-Ориентированного Программирования на языке Python. Классы и объекты

LEARN. GROW. SUCCEED.

 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 #26

Основы Объектно-Ориентированного программирования на языке Python. Классы и объекты

Цель работы

Изучить основы методологии объектно-ориентированного программирования на языке Python; приобрести объектное мышление и научиться проектировать и описывать классы соответствующей предметной области; на базе описанных классов создавать и инициализировать объекты; практически закрепить полученные знания при решении соответствующих задач.

Общее задание

Необходимо решить задачу с использованием методологии ООП. Для чего необходимо подобрать самостоятельно соответствующую проблемную (предметную/доменную) область, которая базируется на объектах и событиях реального мира (примеры соответствующих предметных областей приведены ниже). Спроектировать классы (пользовательские типы данных) в языке Python для программного представления данных объектов и основной логики будущей программной системы. Программа должна решать, как минимум, два полезных действия и иметь дополнительно следующие вещи:

- не менее 3 разнообразных классов предметной области;
- не менее 5 атрибутов (состояния) в каждом соответствующем классе-сущности;
- не менее 3 методов (поведения) в соответствующих функциональных классах для реализации полезной бизнес-логики.

Требования к выполнению задания

1) Необходимо спроектировать UML-диаграмму взаимодействия классов и объектов создаваемой программной системы. На базе данной 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 снабжены 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

Пример предметной области

Автотранспорт *(Automobile transports)*. Определить иерархию автотранспорта соответствующей предметной области (– это могут быть только самолёты или поезда, автобусы или машины, а может и всё вперемешку, Ваши фантазии никто не ограничивает). Создать автопарк (стоянку, аэропорт, вокзала и т.д.). Подсчитать какие-нибудь общие характеристики всех объектов иерархии предметной области или специфические характеристики отдельно взятых объектов иерархии: стоимость всего автопарка машин, общую пассажировместимость, общее количество перевозимого груза, ...

Дополнительно можно осуществить поиск соответствующего объекта по существенной характеристике (к примеру, самую дорогую (дешёвую) машину, или самый длинный (короткий) поезд, самый пассажировместимый самолёт и т.д.). Можно также реализовать сортировку объектов доменной области.

Что нужно запомнить (краткие тезисы)

1. **ООП** использует в качестве базовых элементов объекты, а не алгоритмы.
2. **Объект** – это понятие, абстракция или любой предмет с чётко очерченными границами, имеющий смысл в контексте рассматриваемой прикладной проблемы. Введение объекта преследует две цели: *понимание прикладной задачи (проблемы)* и *введение основы для реализации на компьютере*.
3. **Главная идея ООП** – *всё состоит из объектов*! Программа, написанная с использованием ООП, состоит из множества объектов, и все эти объекты взаимодействуют между собой посредством посылке (передачи) сообщений друг другу. ООП и реальный мир не могут существовать раздельно!
4. **Программные объекты** могут представлять собой объекты реального мира или быть полностью абстрактными объектами, которые могут существовать только в рамках программы. ***Гради Буч*** (создатель унифицированного языка моделирования UML) даёт следующее определение объекта: «***Объект – это мыслимая или реальная сущность, обладающая характерным поведением, отличительными характеристиками и являющая важной в предметной области***».
5. **Каждый объект** имеет состояние, обладает некоторым хорошо определённым поведением и уникальной идентичностью.
6. **Состояние (*state*)** (синонимы: параметры, аспекты, характеристики, свойства, атрибуты, ...) – совокупный результат поведения объекта: одно из стабильных условий, в которых объект может существовать, охарактеризованных количественно; в любой конкретный момент времени состояние объекта включает в себя перечень параметров объекта и текущее значение этих параметров. Состояние объекта в Python реализуется с помощью описания полей класса.
7. **Поведение (*behavior*)** – действия и реакции объекта, выраженные в терминах передачи сообщений и изменения состояния; видимая извне и воспроизводимая активность объекта. Поведение объекта в Python реализуется с помощью описания методов класса.
8. **Уникальность (*identity*)** – эта природа объекта; то, что отличает один объект от других. В машинном представлении уникальность объекта – это адрес размещения объекта в памяти. Следовательно, уникальность объекта состоит в

том, что всегда можно определить, указывают две ссылки на один и тот же объект, или на разные объекты.

9. Чтобы создать программный объект или группу объектов необходимо вначале описать где-то его(их) характеристики и шаблон поведения. Для этих целей в ООП существуют классы (*classes*). Формально, **класс** – это шаблон поведения объектов определённого типа с определёнными параметрам, которые описывают состояние объекта.
10. **Все экземпляры** (объекты) одного и того же класса имеют один и тот же набор характеристик (значение которых может быть различным у разных объектов) и общее поведение (все объекты одинаково реагируют на одинаковые сообщения).

16. **Правила описания класса:**
    a. класс должен быть максимально простым, настолько простым, насколько это возможно;
    b. класс должен отвечать только за ту задачу, которая непосредственно на него возложена (класс должен описывать только одну группу объектов);
    c. название класса – очень важная вещь, оно должно быть простым и понятным (в идеале, имя класса должно говорить о том, за что он отвечает);
    d. название класса обычно называется именем существительным и записывается с заглавной буквы;
    e. классы должны быть самодостаточные (слабосвязанные), т.е. не зависеть от реализации других классов.
17. **Правила описания полей:**
    a. название поля должно однозначно отражать его содержимое;
    b. название поля обычно называется именем существительным и записывается с маленькой буквы;
    c. доступ к полю обычно должно осуществляться через интерфейсную часть объекта или класса.
18. **Правила описания методов:**
    a. название метода должно однозначно отражать, что он выполняет;
    b. название метода обычно называется глаголом и записывается с маленькой буквы;
    c. содержимое метода должно помещаться на экран монитора;
    d. <u>один метод – одно действие!!!;</u>

19. ООП помогает справиться со следующими проблемами:
    - ✓ *уменьшение* ***сложности*** программного обеспечения (ПО);
    - ✓ *увеличение* ***производительности*** *труда программистов и* ***скорости*** разработки ПО;
    - ✓ *повышение* ***надёжности*** ПО;
    - ✓ обеспечение возможности ***модификации*** отдельных компонентов ПО без изменения остальных его частей;
    - ✓ обеспечение возможности ***повторного использования*** отдельных компонентов ПО.

Примеры предметных областей

2. **Цветочница или магазин цветов *(Flower Shop)*.** В магазине цветов можно собрать букет из соответствующих цветов. Необходимо определить вес букета, его стоимость и самый дорогой/недорогой цветок (или цветы, если они одинаковы по стоимости).
3. **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.
4. **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.
5. **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.
6. **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.
7. **Шеф-повар *(Chef)*.** Необходимо приготовить овощной салат и определить вес салата и его калорийность, а также найти самый калорийных овощ в салате.
8. **Шеф-повар *(Chef)*.** Необходимо приготовить фруктовый салат и определить вес салата и его калорийность, а также найти самый калорийных фрукт в салате.
9. **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.

10. **Шеф-повар *(Chef)*.** Необходимо приготовить основное блюдо (или десерт) определить его вес и калорийность, а также найти самый калорийных ингредиент в блюде (десерте).
11. **Игровая комната *(Game room)*.** Есть игровая комната с соответствующими игрушками. Необходимо провести подсчёт стоимости всех игрушек, их общий вес и найти самую дорогую игрушку.
12. **Таксопарк *(Taxi Station)*.** Есть таксопарк, который состоит из соответствующего автотранспорта. Необходимо подсчитать стоимость автопарка и найти самый дорогой автотранспорт по тарифам поездки.
13. **Авиакомпания *(Airline)*.** Есть авиакомпания, которая состоит из соответствующих самолётов. Необходимо подсчитать общее количество мест для пассажиров и грузоподъемность всех самолётов авиакомпании, а также найти самый вместительный самолёт авиакомпании.
14. **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.
15. **Звукозапись *(Sound Recording)*.** В звукозаписывающей студии можно записать на диск сборку музыки. Необходимо подсчитать продолжительность всего диска и найти самую короткую (длинную) песню.
16. **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.
17. **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.
18. **Железнодорожный транспорт *(Railway Transport)*.** Имеются поезда, которые состоят из соответствующих вагонов. Необходимо подсчитать общую длину конкретного поезда, а также найти самый длинный (короткий) вагон.

19. **Автосалон *(Car Center)*.** Есть автосалон, который состоит из соответствующих машин. Необходимо подсчитать общую стоимость машин автосалона и найти самую дорогую (недорогую) машину.
20. **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.
21. **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.
22. **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.
23. **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 объёма и на какую общую сумму.
24. **Рыцарь (Knight).** Есть рыцаря, который экипирован соответствующей амуницией. Необходимо определить общую стоимость всей амуниции, а также найти самую дорогую (недорогую) амуницию рыцаря.
25. **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.
26. **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.

27. **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.
28. **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 (менее) прожорливых животных зоопарка.
29. **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).
30. **Компания по разработки программного обеспечения, IT-компания** (***Software Company***). Есть IT-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.
31. **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.
32. **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.

33. **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).
34. **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.
35. **Тюрьма (*Jail/Prison*).** Есть тюрьма, в которой содержаться заключённые, осужденные 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 заключённого.
36. **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).
37. **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.

38. **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.
39. **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.
40. **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 (столик) с максимальной (минимальной) суммой заказа.
41. **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.

Пример выполнения задания

Предметной (проблемной) областью является выставление оценок студентам преподавателем в университете и подсчёт успеваемости всех студентов.

Пункты решения поставленной задачи:

1) Выделим из предметной области все существительные и глаголы (действия), проанализируем их и попытаемся на базе них описать классы-сущности и функциональные классы с выделением в них существенных характеристик с точки зрения нашей будущей системы. Получаем:
   - ✓ существительные: оценка, преподаватель, университет, успеваемость, студент
   - ✓ действия: выставлять оценки, подсчитывать успеваемость
2) После анализа предметной области спроектируем UML-диаграмму классов и их взаимосвязь друг с другом (смотрите рисунок 1):
   - ✓ студент *(Student)* – класс для описания студента и его успеваемости (класс-сущность, на базе которого будут создаваться далее бизнес объекты, а с этими объектами уже будет работать основная бизнес логика приложения);
   - ✓ преподаватель *(Teacher)* – класс для описания состояния и поведения объекта преподаватель (общий класс, в котором сосредоточена некоторая логика выставления оценок студентам и характеристики преподавателя (имя, стаж работы, и т.д.);
   - ✓ управляющий *(University Manager)* – класс, в котором описывается основная бизнес логика приложения – вычисление успеваемости студентов (функциональный класс);
   - ✓ Всевышний *(God)* – утилитный (дополнительный) класс для создания списка студентов, а также представления данного списка в строковом варианте для вывода на консоль (класс необязательный для доменной области, но облегчает её тестирование).

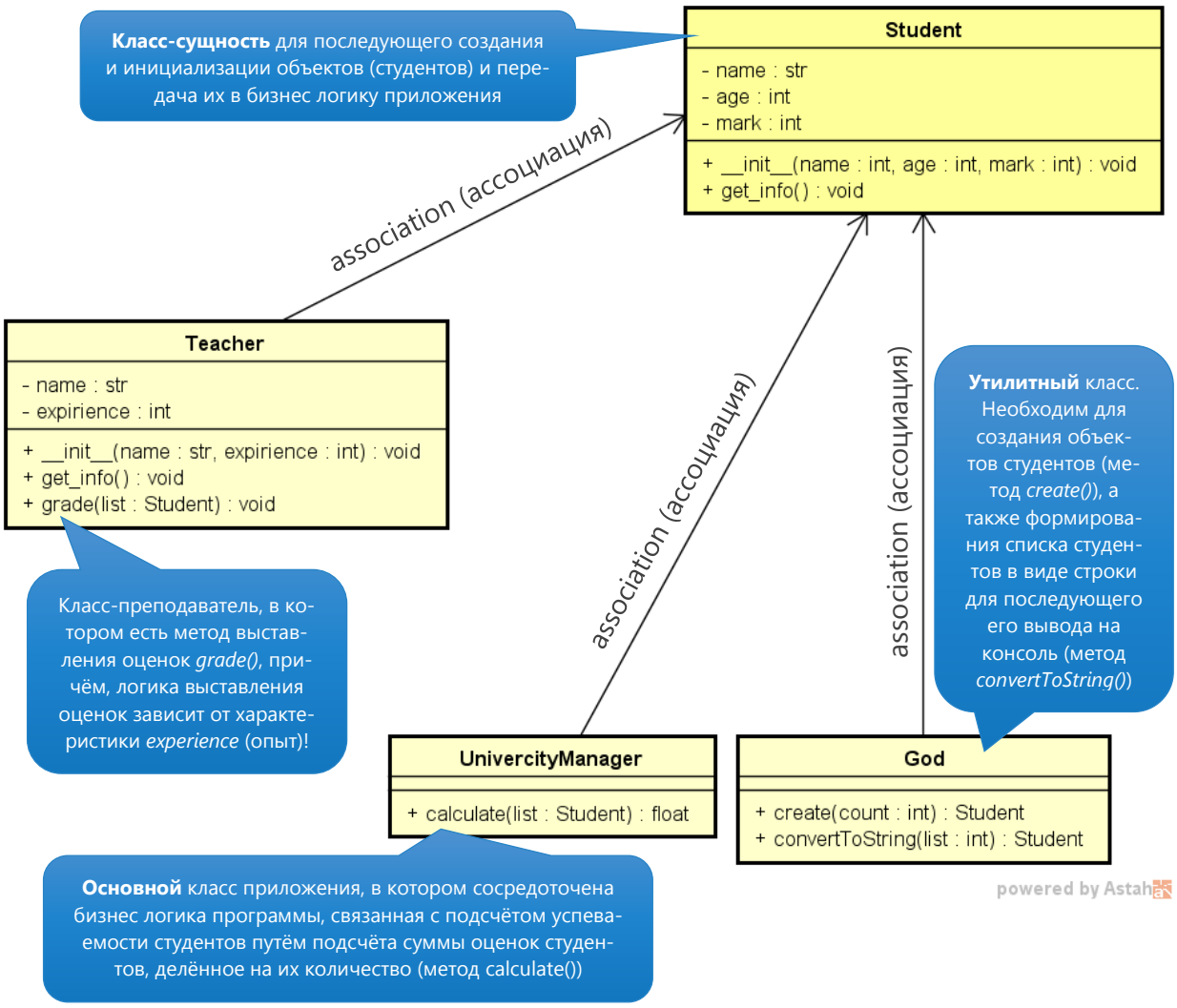

Рисунок 1 – UML-диаграмма классов заданной предметной (проблемной) области

3) На базе приведённой UML-диаграммы разработаем соответствующие классы проблемной области, причём код описания соответствующих классов разместим в отдельных одноимённых модулях (код исходных классов приведён на рисунках 2-5):

```
class Student:
    """ class defines student's information """

    def __init__(self, name, age, mark=4):
        self.name = name
        self.age = age
        self.mark = mark

    def get_info(self):
        return (self.name +
                "(age = " + str(self.age) +
                ", mark = " + str(self.mark) + ")")
```

Специальный метод класса – **конструктор**, который служит для первоначальной инициализации параметров (состояния) создаваемого объекта

Динамический метод для вывода строковых данных о текущем состоянии объекта

Рисунок 2 – Исходный код класса-сущности *Student*

```
import random
from student import Student


class Teacher:
    """class define teacher's information and logic"""

    def __init__(self, name, experience=1):
        self.name = name
        self.experience = experience

    def get_info(self):
        return (self.name + "("
                + str(self.experience) + ")")

    def grade(self, list_of_student):
        MIN_MARK = 4
        MAX_MARK = (100 - self.experience) // 10
        for st in list_of_student:
            if isinstance(st, Student):
                st.mark = random.randint(MIN_MARK, MAX_MARK)
```

Импорт соответствующих модулей. Обратите внимание на порядок их описания: вначале идут всегда импортирование всего модуля, а затем только импортирование конкретных компонентов!

Конструктор класса Teacher для первоначальной инициализации параметров (состояния) создаваемого объекта

Динамический метод для выставления оценок студентам. Обратите внимание, обычно все динамические методы хоть как-то используют состояние объекта для своей работы!

Рисунок 3 – Исходный код общего класса *Teacher*

```python
import random
from student import Student


class God:
    """ functional class for working with list of students"""

    START_ALPHABET_WITH_UPPER_LETTER = 65
    END_ALPHABET_WITH_UPPER_LETTER = 91

    def create(self, count):
        names = ["Alexander", "Pavel", "Artyom", "Michael",
                 "Olya", "Nastya", "Kirill", "Stas", "Nikita",
                 "Oleg", "Max", "Ilya", "Sergey", "Alexey"]

        list_of_student = []

        for i in range(count + 1):
            name = random.choice(names)

            name += " " + chr(random.randint(
                God.START_ALPHABET_WITH_UPPER_LETTER,
                God.END_ALPHABET_WITH_UPPER_LETTER)) + "."

            age = random.randint(17, 19)

            student = Student(name, age)

            list_of_student.append(student)

        return list_of_student

    def convert_to_string(self, list_of_student):
        string = "List of student:\n"

        for student in list_of_student:
            if isinstance(student, Student):
                string += str(student.get_info()) + "\n"

        return string
```

Импорт соответствующих модулей

Задание диапазона значений Unicode-символов заглавных букв английского алфавита

Динамический метод для создания списка студентов

Формирования имени и фамилии студента

Создание объекта «Студент» и его инициализация с помощью конструктора

Добавление объекта «Студент» в общий список

Динамический метод для формирования строкового представления списка студентов

Рисунок 4 – Исходный код утилитного (вспомогательного) класса *God*

```
from student import Student


class Manager:
    """ functional class of business logic"""

    def calculate(self, list_of_student):
        avg = 0
        count = 0
        for st in list_of_student:
            if isinstance(st, Student):
                avg += st.mark
                count += 1

        return float(avg / count)
```

Описание класса бизнес-логики приложения

Динамический метод подсчёта успеваемости студентов (высчитывает среднюю оценку)

Функция ***isinstance()*** используется специально для того, чтобы брать оценку именно у объекта студента, т.к. в списке могут находится также и объекты других классов

Рисунок 5 – Исходный код функционального класса *Manager*

```
from god import God
from teacher import Teacher
from manager import Manager


def main():
    god = God()

    students_list = god.create(10)

    print(god.convert_to_string(students_list))

    teacher = Teacher("Victor Victorovich")

    teacher.grade(students_list)

    print(god.convert_to_string(students_list))

    manager = Manager()

    avg = manager.calculate(students_list)

    print("Avg: %.1f" % avg)


if __name__ == "__main__":
    main()
```

Создание объекта «Создатель»

Посылка сообщения для создания списка из десяти студентов объекту «Создатель»

Посылка сообщения для вывода на консоль списка студентов объекту «Создатель»

Создание объекта «Преподаватель»

Посылка сообщения для выставления оценок студентам объекту «Преподаватель»

Создание объекта «Менеджер»

Посылка сообщения для высчитывания успеваемости студентов объекту «Менеджер»

Рисунок 6 – Стартовый (тестовый) модуль main

4) Для тестирования разработанной модели поведения системы создадим ещё один модуль *main* (см. рис. 6). В нём опишем функцию *main()*, в которой смоделируем создание списка студентов, выставление оценок преподавателем и подсчёт успеваемости студентов, а также вывода списка и результата подсчёта успеваемости в консоль (терминал).

5) Результат работы разработанной программной системы представлен на рисунке 7.

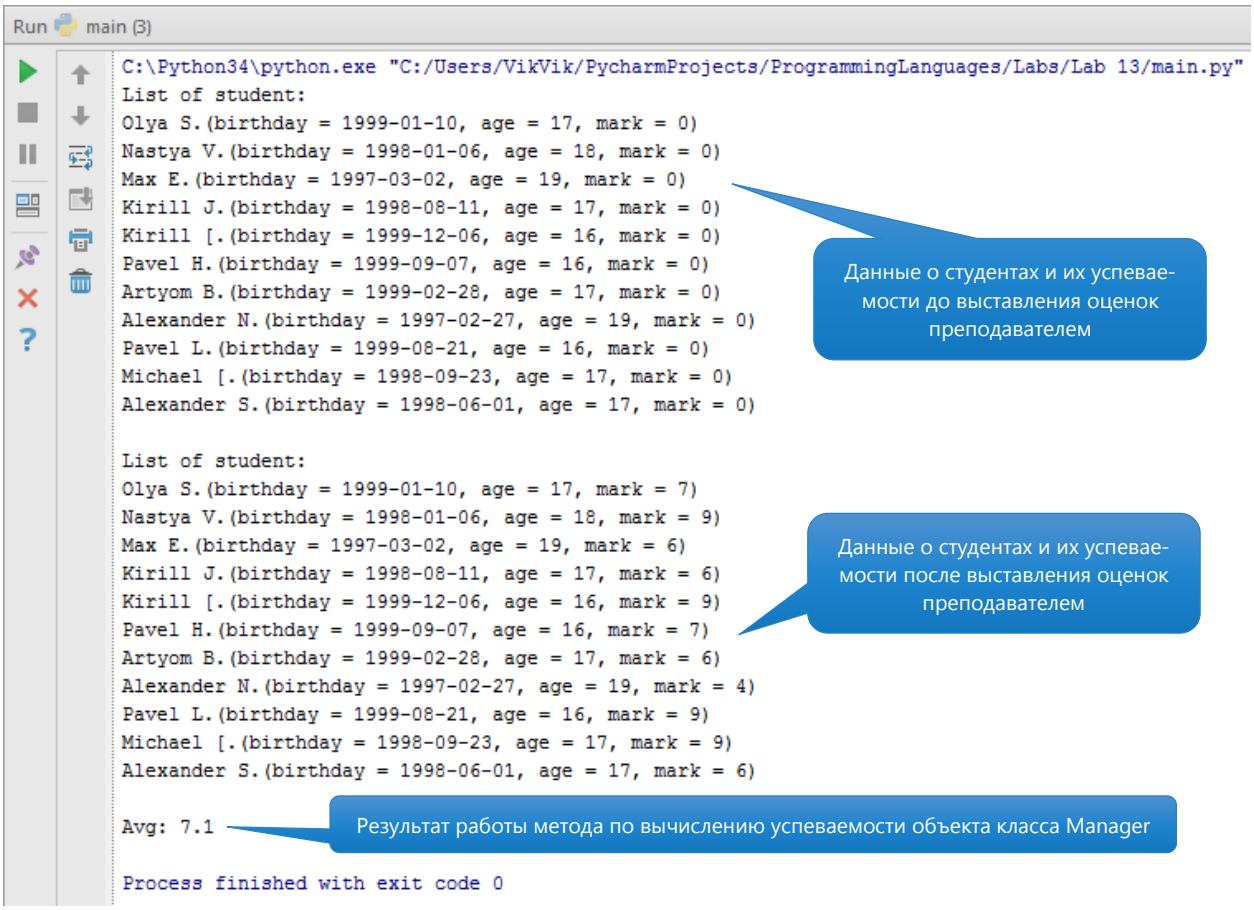

Рисунок 7 – Результат тестирования работы программной системы «Университет»

Контрольные вопросы

1. Что такое ООП? Опишите базовые концепции, которые лежат в основе данной методологии программирования.
2. Для чего было создано ООП? На каких принципах базируется ООП? Опишите их основные идеи?
3. Приведите преимущества и недостатки объектно-ориентированного подхода.
4. Что такое объект и чем характеризуются объекты в ООП?
5. Что такое класс и зачем он нужен? Приведите общее определение класса в ООП. Какая разница между классом и объектом в ООП? Какие разновидности классов существуют в ООП?
6. Как описывается класс в языке Python?
7. Каким способом можно создать объект (экземпляр класса) в языке Python?
8. Как можно обратиться к полям и методам объекта? В чём состоит особенность первого аргумента в методах класса?
9. Какой стандартный метод при описании класса необходимо переопределить, чтобы гарантировать, что в тех ситуациях, где требуется строковое представление объекта данного класса, система автоматических вызывала данный метод у ссылочной переменной?
10. Что такое UML? Как с помощью UML можно описать классы и их взаимосвязь друг с другом (т.е. описать UML-диаграмму классов)?