Сборник заданий для семинарских занятий
по курсу
«Объектно-ориентированное программирование на Python»

Содержание

1 Общие сведения

Сборник содержит задания для семинарских занятий по курсу «Объектно-ориентированное программирование на Python» (32 часа).

Задачник находится в процессе наполнения и новые задания появляются перед проведением нового семинара.

Возможна сдача другого кода (например, выполненного в ходе проектной деятельности), еслои они полностью покрывают материал семинара.

2 Задания

2.1 Семинар «Правила формирования класса для программирования в IDE PyCharm. Отработка навыков создания простых классов и объектов класса» (2 очных часа)

В ходе работы создайте 5 классов с соответствующими методами, описанными в индивидуальном задании. Предполагается, что пользователь класса не имеет права обращаться к свойствам напрямую (соблюдая принцип инкапсуляции), а должен использовать методы. Важно: в задании не всегда указаны все необходимые методы и свойства, при необходимости вам надо самостоятельно их добавить. Продемонстрируйте работоспособность всех методов (из задания) посредством создания запускаемых файлов, где осуществляется вызов методов для разных ситуаций (без ручного ввода, но с выводом результатов в консоль). Каждый класс должен сохраняться в отдельном исходном файле. Необходимо соблюдать все стандартные требования к качеству кода (отступы, именования переменных, классов, методов, проверка корректности входных данных). Для каждого класса создайте отдельный запускаемый файл для проверки всех его методов (допускается использование других классов в этих тестах).

Все предлагаемые классы в заданиях упрощенные; для использования в production-окружении они требуют серьезной доработки. Суть задания — в отработке базовых навыков, а не в идеальном моделировании предложенных ситуаций.

Для сдачи работы будьте готовы пояснить или аналогично заданию модифицировать любую часть кода, а также ответить на вопросы:

1. Кратко опишите парадигму объектно-ориентированного программирования (ООП).
2. Что такое класс в парадигме ООП?
3. Что такое объект (экземпляр) в парадигме ООП?
4. Что обозначает свойство инкапсуляции в парадигме ООП?
5. Синтаксис классов в Python (в рамках выполненной работы), создание и работа с объектами в Python.

При выполнении задания предполагается самое простое базовое описание классов, соответствующее следующему примеру (вы можете использовать то, что вы ЗНАЕТЕ дополнительно, но это остается на ваше усмотрение):

Если вы нашли в задачнике ошибки, опечатки и другие недостатки, то вы можете сделать pull-request.

```
class Worker:
    def set_last_name(self, last_name):
        self.last_name = last_name

    def print_last_name(self):
        print (f"Фамилия: {self.last_name}")

    def get_last_name(self):
        return last_name

worker = Worker()
worker.set_last_name(self,"Иванов")
worker.print_last_name()
print(worker.get_last_name())
```

Срок сдачи работы (начала сдачи): следующее занятие после его выдачи. В последующие сроки оценка будет снижаться (при отсутствии оправдывающих документов).

1. **Описание ситуации:** Рассмотрим работу грузовой железнодорожной станции. На станции есть несколько путей, по которым поезда могут прибывать и отправляться. Каждый путь имеет свой номер и может вместить несколько поездов. Поезда формируются из вагонов, каждый из которых может перевозить разные грузы. Работники станции отвечают за диспетчерское управление маневровыми локомотивами, осмотр вагонов, выполнение погрузочно-разгрузочных работ, прием груза к перевозке, ремонт путей, обеспечение безопасности и т.п. Они используют радиостанции для связи друг с другом и для отслеживания положения поездов и передвижения вагонов.

   Создаваемые классы: ‘Путь‘, ‘Поезд‘, ‘Вагон‘, ‘Станция‘, ‘РаботникСтанции‘.

   Для классов реализовать следующие простые методы (ниже приведен не исчерпывающий список методов; для демонстрации работы классов вам потребуются дополнительные методы, позволяющие отследить состояние объектов), используя для хранения данных списки (‘[]‘) Python:

   (a) **Путь:** добавить поезд на путь, убрать поезд с пути, получить список поездов на конкретном пути.

   (b) **Поезд:** прицепить вагон к поезду, отцепить вагон от поезда, получить (распечатать) список вагонов в поезде, вывести информацию о грузе в поезде.

   (c) **Вагон:** добавить номер поезда, в который включался конкретный вагон, удалить номер поезда из истории, отобразить историю поездов для конкретного вагона.

   (d) **РаботникСтанции:** класс, представляющий отдельного работника на станции, имеющий идентификатор, информацию о персональной радиостанции, список закрепленных за ним поездов для осмотра, ФИО, должность.

   (e) **Станция:** добавить станционный путь, добавить поезд на станцию, нанять работника станции, вывести информацию о всех путях, поездах, работниках, удалить путь, удалить поезд, уволить работника.

2. **Описание ситуации:** Рассмотрим работу крупного логистического терминала для обработки грузовых автомобилей. На терминале есть несколько доков (рамп), куда фуры прибывают для проведения погрузочно-разгрузочных работ. Каждый док имеет свой номер и может одновременно обслуживать одну машину. Грузовики перевозят паллеты, каждая из которых содержит определенный товар. Сотрудники терминала отвечают за прием грузовиков, управление погрузочной техникой, проверку сопроводительных документов, приемку и отгрузку товара, а также техническое обслуживание доков. Они используют портативные рации для координации действий и отслеживания статуса обработки автомобилей.

   Создаваемые классы: ‘Док‘, ‘Грузовик‘, ‘Паллета‘, ‘Терминал‘, ‘Сотрудник‘.

   Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

   (a) **Док:** занять док конкретным грузовиком, освободить док, получить информацию о грузовике, который сейчас находится на доке.

   (b) **Грузовик:** добавить паллету в грузовик, выгрузить паллету из грузовика, получить (распечатать) список паллет в грузовике, вывести информацию о товарах в грузовике.

   (c) **Паллета:** добавить номер грузовика, в который загружалась конкретная паллета, удалить номер грузовика из истории, отобразить историю перевозок (номера грузовиков) для конкретной паллеты.

(d) **Сотрудник:** класс, представляющий отдельного сотрудника терминала, имеющий идентификатор, номер рации, список доков, за которые он отвечает, ФИО, должность.

(e) **Терминал:** добавить новый док на терминале, зарегистрировать прибытие грузовика, нанять нового сотрудника, вывести список всех доков, грузовиков на территории, сотрудников, удалить док, удалить грузовик, уволить сотрудника.

3. **Описание ситуации:** Рассмотрим работу аэропорта. В аэропорту есть несколько взлетно-посадочных полос (ВПП), которые принимают и отправляют рейсы. Каждая ВПП имеет свой номер, длину и статус доступности. Самолеты перевозят пассажиров и их ручную кладь, размещенную в салоне. Авиадиспетчеры управляют движением самолетов, назначают полосы для взлета и посадки, следят за воздушной обстановкой и координируют действия с помощью радиосвязи.

   Создаваемые классы: ‘ВПП‘, ‘Самолет‘, ‘Пассажир‘, ‘Аэропорт‘, ‘Авиадиспетчер‘.

   Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

   (a) **ВПП:** занять полосу для взлета/посадки, освободить полосу, получить список рейсов, использовавших полосу.

   (b) **Самолет:** добавить пассажира на борт (включая вес его ручной клади), высадить пассажира, получить (распечатать) список пассажиров на борту, рассчитать общий вес ручной клади.

   (c) **Пассажир:** добавить рейс в историю перелетов пассажира, удалить рейс из истории (ошибка бронирования), отобразить всю историю перелетов.

   (d) **Авиадиспетчер:** класс, представляющий диспетчера, имеющий идентификатор, рабочую частоту, график работы (список интервалов времени в сутках), ФИО.

   (e) **Аэропорт:** добавить новую ВПП, зарегистрировать прибытие самолета, нанять диспетчера, вывести список всех ВПП, самолетов в аэропорту, диспетчеров, удалить ВПП (на ремонт), списать самолет, уволить диспетчера.

4. **Описание ситуации:** Рассмотрим работу речного порта. В порту есть несколько причалов для швартовки грузовых барж и буксиров. Каждый причал имеет уникальный номер и максимальную глубину, определяющую осадку судов, которые могут к нему подойти. Баржи перевозят контейнеры с различными грузами. Их характеризуют вес судна, максимальная грузоподъемность и осадка (как без груза, так и с максимальным грузом). Портовые рабочие отвечают за швартовку судов, управление портовыми кранами для погрузки/разгрузки контейнеров, оформление документов и поддержание порядка на территории.

   Создаваемые классы: ‘Причал‘, ‘Баржа‘, ‘Контейнер‘, ‘Порт‘, ‘ПортовыйРабочий‘.

   Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

   (a) **Причал:** пришвартовать баржу к причалу, отшвартовать баржу, получить список барж, находящихся у причала.

   (b) **Баржа:** загрузить контейнер на баржу (с указанием веса контейнера), разгрузить контейнер с баржи, получить (распечатать) список контейнеров на барже, рассчитать текущую осадку судна (предполагается линейная зависимость осадки от суммарного веса груза и баржи).

(c) **Контейнер:** добавить номер баржи, на которую погрузили контейнер, удалить номер баржи, отобразить историю перемещений контейнера между баржами.

(d) **ПортовыйРабочий:** класс, представляющий рабочего, имеющий идентификатор, допуск к работе с краном, список закрепленных причалов, ФИО, должность.

(e) **Порт:** ввести новый причал в эксплуатацию, принять баржу в акваторию порта, принять на работу рабочего, вывести список причалов, барж в акватории, рабочих, списать причал, отправить баржу, уволить рабочего.

5. **Описание ситуации:** Рассмотрим работу автобусного парка. В парке есть несколько маршрутов, которые обслуживаются автобусами. Каждый маршрут имеет номер и список остановок. Автобусы имеют государственный номер, количество мест и текущий пробег. Водители закреплены за автобусами и маршрутами. Диспетчеры автопарка составляют расписание, следят за выходами автобусов на линию, учетом пробега и техническим состоянием.

   Создаваемые классы: ‘Маршрут‘, ‘Автобус‘, ‘Остановка‘, ‘Автопарк‘, ‘Водитель‘.

   Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

   (a) **Маршрут:** добавить остановку в маршрут, удалить остановку из маршрута, получить список всех остановок на маршруте.

   (b) **Автобус:** назначить автобус на маршрут, снять с маршрута, увеличить пробег на заданное значение, получить текущий пробег.

   (c) **Остановка:** добавить маршрут, проходящий через остановку, удалить маршрут, отобразить список всех маршрутов, проходящих через данную остановку.

   (d) **Водитель:** класс, представляющий водителя, имеющий идентификатор, права категории, закрепленный автобус, ФИО, график работы.

   (e) **Автопарк:** добавить новый маршрут, приобрести новый автобус, принять на работу водителя, вывести список маршрутов, автобусов (с указанием их состояния), водителей, списать автобус, уволить водителя.

6. **Описание ситуации:** Рассмотрим работу метрополитена. В метро есть линии, состоящие из станций и тоннелей между ними. Составы из вагонов перемещаются по линиям. Каждая станция имеет название и может быть точкой пересадки на другие линии. Машинисты управляют поездами. Дежурные по станции следят за порядком на платформах и работой оборудования. Управление метрополитеном координирует движение составов.

   Создаваемые классы: ‘ЛинияМетро‘, ‘ПоездМетро‘, ‘Станция‘, ‘УправлениеМетрополитеном‘, ‘Машинист‘.

   Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

   (a) **ЛинияМетро:** добавить станцию на линию, получить список станций на линии, получить список поездов на линии.

   (b) **ПоездМетро:** добавить вагон в состав, отцепить вагон, назначить машиниста на поезд.

   (c) **Станция:** добавить линию, проходящую через станцию (для моделирования пересадочных узлов), получить список линий на станции.

(d) **Машинист:** класс, представляющий машиниста, имеющий идентификатор, допуск к управлению, закрепленный поезд, ФИО, стаж.

(e) **УправлениеМетрополитеном:** открыть новую линию, ввести новый поезд в эксплуатацию, принять на работу машиниста, вывести список линий, поездов (в депо и на линиях), машинистов, закрыть линию на техобслуживание, списать поезд, вывести полную схему метро (в текстовом виде).

7. **Описание ситуации:** Рассмотрим работу службы доставки пиццы. В службе есть несколько филиалов. Каждый филиал обслуживает определенный район и имеет курьеров. Заказы формируются из позиций меню. Курьеры используют скутеры для доставки. Менеджеры филиалов принимают заказы, назначают курьеров и следят за выполнением заказов.

   Создаваемые классы: ‘Филиал‘, ‘Заказ‘, ‘Курьер‘, ‘Скутер‘, ‘Менеджер‘.

   Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

   (a) **Филиал:** добавить курьера в филиал, уволить курьера, получить список активных заказов филиала.

   (b) **Заказ:** добавить позицию в заказ (название + цена), удалить позицию, рассчитать стоимость заказа, изменить статус заказа (принят, готовится, в пути, доставлен).

   (c) **Курьер:** назначить заказ курьеру, завершить доставку заказа, получить список доставленных заказов за смену, закрепить скутер за курьером.

   (d) **Менеджер:** класс, представляющий менеджера, имеющий идентификатор, закрепленный филиал, ФИО, смену.

   (e) **Скутер:** отправить на зарядку, вернуть в строй, увеличить пробег, получить текущий пробег.

   Описание ситуации: Рассмотрим работу трамвайного депо. В депо есть несколько маршрутов, обслуживаемых трамвайными вагонами. Каждый трамвайный вагон имеет бортовой номер, вместимость и текущий пробег. Маршруты состоят из остановок и имеют определенный график движения. Водители трамваев закреплены за конкретными вагонами и маршрутами. Диспетчеры управляют выпуском трамваев на линию и ведут учет технического состояния.

   Создаваемые классы: Маршрут, Трамвай, Остановка, Депо, Водитель.

   Для классов реализовать следующие простые методы, используя для хранения данных списки ([]) Python:

   (a) **Маршрут:** добавить остановку в маршрут, удалить остановку из маршрута, получить список всех остановок на маршруте.

   (b) **Трамвай:** назначить трамвай на маршрут, снять с маршрута, увеличить пробег на заданное значение, получить текущий пробег.

   (c) **Остановка:** добавить маршрут, проходящий через остановку, удалить маршрут, отобразить список всех маршрутов, проходящих через данную остановку.

   (d) **Водитель:** класс, представляющий водителя, имеющий идентификатор, права категории, закрепленный трамвай, ФИО, график работы.

(e) **Депо:** добавить новый маршрут, принять новый трамвай в депо, принять на работу водителя, выполнить вывод списка маршрутов, трамваев (с указанием их состояния), водителей, списать трамвай, уволить водителя.

8. **Описание ситуации:** Рассмотрим работу морского порта для приёма пассажирских паромов. В порту есть несколько причалов, каждый из которых обслуживает один паром за раз. Паромы перевозят пассажиров и автомобили. Пассажиры покупают билеты, автомобили записываются в список грузовой палубы. Сотрудники порта координируют погрузку, проверку билетов и безопасность. **Создаваемые классы:** Причал, Паром, Пассажир, Автомобиль, СотрудникПорта.

   (a) **Причал:** пришвартовать паром, освободить причал, получить информацию о пароме у причала.

   (b) **Паром:** добавить пассажира, добавить автомобиль, высадить пассажира, выгрузить автомобиль.

   (c) **Пассажир:** добавить рейс в историю поездок, удалить рейс из истории, вывести историю поездок.

   (d) **Автомобиль:** зарегистрировать номер парома, удалить номер парома, вывести историю перевозок.

   (e) **СотрудникПорта:** идентификатор, должность, ФИО, список закреплённых причалов.

9. **Описание ситуации:** Рассмотрим работу пригородной электрички. В системе есть станции, между которыми курсируют электрички. У каждой электрички есть номер, список вагонов и машинист. Пассажиры покупают билеты и занимают места в вагонах. Диспетчеры контролируют движение электричек. **Создаваемые классы:** Станция, Электричка, Вагон, Пассажир, Диспетчер.

   (a) **Станция:** принять электричку, отправить электричку, вывести список электричек на станции.

   (b) **Электричка:** добавить вагон, отцепить вагон, получить список вагонов.

   (c) **Вагон:** посадить пассажира, высадить пассажира, вывести список пассажиров.

   (d) **Пассажир:** добавить поездку в историю, удалить поездку, показать историю поездок.

   (e) **Диспетчер:** идентификатор, ФИО, рабочая смена, список контролируемых электричек.

10. **Описание ситуации:** Рассмотрим работу таксопарка. В таксопарке есть автомобили, водители и диспетчеры. Автомобиль закрепляется за водителем. Диспетчеры принимают заказы и назначают их водителям. Пассажиры совершают поездки. **Создаваемые классы:** Таксопарк, Автомобиль, Водитель, Заказ, Диспетчер.

    (a) **Таксопарк:** добавить автомобиль, принять водителя, вывести список машин и водителей, уволить водителя.

    (b) **Автомобиль:** назначить водителя, снять водителя, увеличить пробег, получить пробег.

    (c) **Водитель:** назначить заказ, завершить заказ, вывести список выполненных заказов.

(d) **Заказ:** назначить пассажира, завершить поездку, вывести информацию о заказе.

(e) **Диспетчер:** идентификатор, ФИО, список назначенных заказов.

11. **Описание ситуации:** Рассмотрим работу грузового аэропорта. Самолёты перевозят контейнеры. В аэропорту есть ангары для хранения самолётов и площадки для погрузки. Работники аэропорта координируют загрузку и выгрузку контейнеров. **Создаваемые классы:** Самолёт, Контейнер, Ангар, РаботникАэропорта, Аэропорт.

    (a) **Самолёт:** загрузить контейнер, выгрузить контейнер, вывести список контейнеров.

    (b) **Контейнер:** добавить номер самолёта, удалить номер самолёта, вывести историю перевозок.

    (c) **Ангар:** принять самолёт, вывести список самолётов, освободить ангар.

    (d) **РаботникАэропорта:** идентификатор, ФИО, должность, список самолётов в обслуживании.

    (e) **Аэропорт:** принять самолёт, убрать самолёт, принять раотника, уволить работника, вывести список самолётов и работников.

12. **Описание ситуации:** Рассмотрим работу велопроката. В прокате есть велосипеды, станции для их хранения, клиенты и сотрудники. Клиенты арендуют велосипеды и возвращают их на станцию. **Создаваемые классы:** Велосипед, СтанцияПроката, Клиент, Сотрудник, Прокат.

    (a) **Велосипед:** выдать в аренду, вернуть на станцию, получить пробег.

    (b) **СтанцияПроката:** добавить велосипед, убрать велосипед, вывести список велосипедов.

    (c) **Клиент:** арендовать велосипед, вернуть велосипед, вывести историю аренд.

    (d) **Сотрудник:** идентификатор, ФИО, должность, список закреплённых станций.

    (e) **Прокат:** добавить станцию, демонтировать станцию, вывести список станций и велосипедов, уволить сотрудника, нанять сотрудника, вывести список сотрудников.

13. **Описание ситуации:** Рассмотрим работу речных теплоходов. У каждого теплохода есть рейсы и список пассажиров. Пассажиры покупают билеты. Работники пристани обслуживают теплоходы. **Создаваемые классы:** Теплоход, Рейс, Пассажир, Пристань, РаботникПристани.

    (a) **Теплоход:** добавить рейс, убрать рейс, вывести список рейсов.

    (b) **Рейс:** добавить пассажира, удалить пассажира, вывести список пассажиров.

    (c) **Пассажир:** добавить рейс в историю, удалить рейс, вывести историю.

    (d) **Пристань:** принять теплоход, отправить теплоход, вывести список теплоходов.

    (e) **РаботникПристани:** идентификатор, ФИО, должность, закреплённые рейсы.

14. **Описание ситуации:** Рассмотрим работу каршеринга. В системе есть автомобили, клиенты и диспетчеры. Автомобили бронируются клиентами и возвращаются после поездки. Диспетчеры контролируют состояние машин. **Создаваемые классы:** Автомобиль, Клиент, Диспетчер, Заказ, Каршеринг.

(a) **Автомобиль:** выдать клиенту, вернуть, увеличить пробег, вывести пробег.

(b) **Клиент:** арендовать автомобиль, завершить аренду, вывести историю аренд.

(c) **Диспетчер:** идентификатор, ФИО, список автомобилей под контролем.

(d) **Заказ:** назначить автомобиль, завершить поездку, вывести данные заказа.

(e) **Каршеринг:** добавить автомобиль, списать автомобиль, добавить клиента, удалить клиента, добавить диспетчера, удалить диспетчера, вывести список клиентов, диспетчеров и машин.

15. **Описание ситуации:** Рассмотрим работу железнодорожного музея. В музее есть экспонаты (локомотивы и вагоны), экскурсии и экскурсоводы. Посетители записываются на экскурсии. **Создаваемые классы:** Экспонат, Экскурсия, Экскурсовод, Посетитель, Музей.

    (a) **Экспонат:** добавить к экскурсии, убрать, вывести список экскурсий.

    (b) **Экскурсия:** записать посетителя, удалить, вывести список посетителей.

    (c) **Экскурсовод:** идентификатор, ФИО, список экскурсий.

    (d) **Посетитель:** записаться на экскурсию, отменить запись, вывести историю.

    (e) **Музей:** добавить экспонат, списать экспонат, добавить экскурсовода, уволить экскурсовода, провести экскурсию, вывести список всех экскурсий и экскурсоводов.

16. **Описание ситуации:** Рассмотрим работу автозаправочной станции. На станции есть топливо, колонки и операторы. Автомобили приезжают заправляться. **Создаваемые классы:** Колонка, Автомобиль, Оператор, Топливо, АЗС.

    (a) **Колонка:** заправить автомобиль, освободить колонку, вывести статус.

    (b) **Автомобиль:** получить заправку, вывести историю заправок.

    (c) **Оператор:** идентификатор, ФИО, список закреплённых колонок.

    (d) **Топливо:** уменьшить количество, увеличить количество, вывести остаток.

    (e) **АЗС:** добавить колонку, нанять оператора, уволить оператора, демонтировать колонку, вывести список машин, операторов и колонок.

17. **Описание ситуации:** Рассмотрим работу сортировочного центра курьерской службы. В центре есть зоны обработки посылок, конвейерные линии и сотрудники. Каждая посылка имеет трек-номер и проходит через несколько этапов обработки. Сотрудники сканируют посылки, сортируют их по направлениям и загружают в транспортировочные контейнеры. Менеджеры контролируют процесс сортировки и работу оборудования.

    Создаваемые классы: ‘ЗонаОбработки‘, ‘Посылка‘, ‘Конвейер‘, ‘СотрудникЦентра‘, ‘СортировочныйЦентр‘.

    Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

    (a) **ЗонаОбработки:** добавить посылку в зону, удалить посылку из зоны, получить список посылок в зоне.

    (b) **Посылка:** добавить статус обработки (принята, сортируется, отправлена), удалить ошибочный статус, отобразить историю статусов обработки.

(c) **Конвейер:** запустить конвейерную ленту, остановить конвейер, добавить посылку на конвейер, снять посылку с конвейера.

(d) **СотрудникЦентра:** класс, представляющий сотрудника, имеющий идентификатор, смену, список закрепленных зон обработки, ФИО, должность.

(e) **СортировочныйЦентр:** добавить новую зону обработки, ввести в эксплуатацию конвейер, нанять сотрудника, вывести список всех зон, конвейеров, сотрудников, удалить зону, вывести из эксплуатации конвейер, уволить сотрудника.

18. **Описание ситуации:** Рассмотрим работу диспетчерской службы городского пассажирского транспорта. Диспетчеры отслеживают движение автобусов, троллейбусов и трамваев на маршрутах, регулируют интервалы движения, фиксируют отклонения от графика. Транспортные средства оснащены GPS-трекерами для передачи местоположения.

    Создаваемые классы: ‘Маршрут‘, ‘ТранспортноеСредство‘, ‘Диспетчер‘, ‘Остановка‘, ‘ДиспетчерскаяСлужба‘.

    Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

    (a) **Маршрут:** добавить транспортное средство на маршрут, снять с маршрута, получить список транспорта на маршруте.

    (b) **ТранспортноеСредство:** обновить местоположение (координаты), получить текущее местоположение, добавить информацию о задержке/опережении графика.

    (c) **Диспетчер:** класс, представляющий диспетчера, имеющий идентификатор, смену, список контролируемых маршрутов, ФИО.

    (d) **Остановка:** добавить маршрут, проходящий через остановку, удалить маршрут, получить список маршрутов на остановке.

    (e) **ДиспетчерскаяСлужба:** добавить новый маршрут, зарегистрировать транспортное средство, нанять диспетчера, вывести информацию о всех маршрутах, транспорте, диспетчерах, удалить маршрут, списать транспорт, уволить диспетчера.

19. **Описание ситуации:** Рассмотрим работу центра технического обслуживания городского транспорта. В центре есть ремонтные зоны для разных видов транспорта, запасы запчастей и бригады механиков. Транспортные средства проходят плановое ТО и внеплановый ремонт.

    Создаваемые классы: ‘РемонтнаяЗона‘, ‘ТранспортноеСредство‘, ‘Запчасть‘, ‘Механик‘, ‘ЦентрТехОбслуживания‘.

    Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

    (a) **РемонтнаяЗона:** поставить транспорт на ремонт, завершить ремонт, получить список транспорта в ремонте.

    (b) **ТранспортноеСредство:** добавить запись о ремонте (дата, вид работ), удалить ошибочную запись, отобразить историю ремонтов.

    (c) **Запчасть:** уменьшить количество на складе, увеличить количество, получить текущий остаток.

(d) **Механик:** класс, представляющий механика, имеющий идентификатор, квалификацию, список закрепленных ремонтных зон, ФИО.

(e) **ЦентрТехОбслуживания:** добавить ремонтную зону, закупить запчасти, нанять механика, вывести информацию о зонах, запчастях, механиках, удалить зону, уволить механика.

20. **Описание ситуации:** Рассмотрим работу логистического центра междугородных автобусных перевозок. Автобусы совершают рейсы между городами, перевозя пассажиров и багаж. Диспетчеры формируют расписание, продают билеты и контролируют отправление автобусов.

21. **Описание ситуации:** Рассмотрим работу центра управления интеллектуальной транспортной системой города. Система включает в себя управление светофорами, камеры видеонаблюдения, датчики транспортного потока. Операторы следят за дорожной ситуацией и оперативно реагируют на инциденты.

    Создаваемые классы: ‘Перекресток‘, ‘Светофор‘, ‘КамераНаблюдения‘, ‘ОператорИТС‘, ‘ЦентрУправления‘.

    Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

    (a) **Перекресток:** добавить светофор к перекрестку, удалить светофор, получить список светофоров на перекрестке.

    (b) **Светофор:** изменить режим работы (красный/желтый/зеленый), получить текущий режим, добавить информацию о неисправности, вывести список неисправностей.

    (c) **КамераНаблюдения:** включить запись, выключить запись, получить статус работы, зафиксировать нарушение ПДД, вывести список нарушений.

    (d) **ОператорИТС:** класс, представляющий оператора, имеющий идентификатор, смену, список контролируемых перекрестков, ФИО.

    (e) **ЦентрУправления:** добавить новый перекресток в систему, установить светофор, установить камеру, нанять оператора, вывести информацию о перекрестках, светофорах, камерах, операторах, удалить перекресток, уволить оператора, снять камеру, снять светофор.

22. **Описание ситуации:** Рассмотрим работу службы эвакуации аварийных транспортных средств. Эвакуаторы дежурят на специальных парковках и выезжают по вызову на места ДТП или поломок. Диспетчеры принимают вызовы и направляют ближайший свободный эвакуатор.

    Создаваемые классы: ‘Эвакуатор‘, ‘Вызов‘, ‘ПарковкаЭвакуаторов‘, ‘ДиспетчерЭвакуации‘, ‘СлужбаЭвакуации‘.

    Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

    (a) **Эвакуатор:** принять вызов, завершить вызов, получить текущий статус (свободен/занят), обновить местоположение.

    (b) **Вызов:** зафиксировать время принятия, время выполнения, получить статус выполнения.

(c) **ПарковкаЭвакуаторов:** принять эвакуатор на парковку, выпустить эвакуатор с парковки, получить список эвакуаторов на парковке.

(d) **ДиспетчерЭвакуации:** класс, представляющий диспетчера, имеющий идентификатор, смену, список обработанных вызовов, ФИО.

(e) **СлужбаЭвакуации:** добавить эвакуатор в парк, списать эвакуатор, нанять диспетчера, вывести информацию о эвакуаторах, вызовах, диспетчерах, уволить диспетчера.

23. **Описание ситуации:** Рассмотрим работу центра контроля коммерческих грузоперевозок. Система отслеживает движение грузовых автомобилей, контролирует соблюдение маршрутов, норм труда водителей и расход топлива. Менеджеры по логистике планируют маршруты и анализируют отчеты.

    Создаваемые классы: ‘ГрузовойАвтомобиль‘, ‘МаршрутПеревозки‘, ‘Водитель‘, ‘Рейс‘, ‘МенеджерЛогистики‘.

    Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

    (a) **ГрузовойАвтомобиль:** начать рейс, завершить рейс, получить текущий статус, зафиксировать расход топлива.

    (b) **МаршрутПеревозки:** добавить точку маршрута (город, склад), удалить точку, получить полный маршрут.

    (c) **Водитель:** класс, представляющий водителя, имеющий идентификатор, права, график работы, ФИО, стаж.

    (d) **Рейс:** закрепить автомобиль за рейсом, закрепить водителя за рейсом, открепить автомобиль, снять водителя, получить информацию о рейсе.

    (e) **МенеджерЛогистики:** класс, представляющий менеджера, имеющий идентификатор, список контролируемых маршрутов, ФИО.

24. **Описание ситуации:** Рассмотрим работу службы парковки аэропорта. На территории аэропорта есть несколько парковочных зон для разных типов транспорта (краткосрочная, долгосрочная, VIP). Операторы контролируют занятость мест, прием оплаты и работу шлагбаумов.

    Создаваемые классы: ‘ПарковочнаяЗона‘, ‘ПарковочноеМесто‘, ‘Автомобиль‘, ‘ОператорПарковки‘, ‘СлужбаПарковки‘.

    Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

    (a) **ПарковочнаяЗона:** добавить парковочное место, удалить место, получить список мест в зоне, получить список всех автомобилей. Так же парковочной зоне соответсвует стоимость часа стоянки.

    (b) **ПарковочноеМесто:** занять место автомобилем, освободить место, получить текущий статус (свободно/занято).

    (c) **Автомобиль:** зафиксировать время въезда, время выезда + рассчитать стоимость парковки (с учетом стоимости часа), получить историю.

    (d) **ОператорПарковки:** класс, представляющий оператора, имеющий идентификатор, смену, список контролируемых зон, ФИО.

(e) **СлужбаПарковки:** добавить новую парковочную зону, нанять оператора, вывести информацию о зонах, местах, операторах, удалить зону, уволить оператора.

25. **Описание ситуации:** Рассмотрим работу центра управления речным судоходством. Диспетчеры следят за движением судов по фарватеру, распределяют шлюзы, контролируют соблюдение графика движения и обеспечивают безопасность судоходства.

    Создаваемые классы: ‘Судно‘, ‘Шлюз‘, ‘Фарватер‘, ‘ДиспетчерСудоходства‘, ‘ЦентрУправления‘.

    Для классов реализовать следующие простые методы, используя для хранения данных списки (‘[]‘) Python:

    (a) **Судно:** начать движение по фарватеру, завершить движение, получить текущее местоположение, зафиксировать прохождение шлюза.

    (b) **Шлюз:** принять судно для шлюзования, завершить шлюзование, получить текущий статус (свободен/занят).

    (c) **Фарватер:** добавить участок фарватера, удалить участок, получить список судов на фарватере.

    (d) **ДиспетчерСудоходства:** класс, представляющий диспетчера, имеющий идентификатор, смену, список контролируемых шлюзов, ФИО.

    (e) **ЦентрУправления:** добавить шлюз в систему, зарегистрировать судно, нанять диспетчера, вывести информацию о шлюзах, фарватерах, судах, диспетчерах, удалить шлюз, уволить диспетчера.

26. **Описание ситуации:** Рассмотрим работу службы технического контроля метрополитена. Инспекторы проверяют состояние путей, тоннелей, подвижного состава и оборудования станций. Дефекты фиксируются в системе для оперативного устранения ремонтными бригадами.

    Создаваемые классы: ‘УчастокПути‘, ‘ПодвижнойСостав‘, ‘Инспектор‘, ‘Дефект‘, ‘СлужбаКонтроля‘.

    Для классов реализовать следующие простые методы, использующие для хранения данных списки (‘[]‘) Python:

    (a) **УчастокПути:** добавить информацию о дефекте, получить список неустраненных дефектов на участке.

    (b) **ПодвижнойСостав:** добавить запись о техническом осмотре, удалить ошибочную запись, отобразить историю осмотров.

    (c) **Инспектор:** класс, представляющий инспектора, имеющий идентификатор, квалификацию, список закрепленных участков, ФИО.

    (d) **Дефект:** зафиксировать время обнаружения, время устранения, получить статус устранения.

    (e) **СлужбаКонтроля:** добавить участок пути в систему, зарегистрировать подвижной состав, нанять инспектора, вывести информацию об участках, составе, инспекторах, дефектах, удалить участок, уволить инспектора, снять с эксплуатации подвижной состав.

27. **Описание ситуации:** Рассмотрим работу центра управления умными светофорами на перекрестках. Умные светофоры адаптивно меняют режим работы в зависимости

от транспортного потока, приоритизируя общественный транспорт и спецтранспорт. Система анализирует данные с датчиков и камер, оптимизируя пропускную способность перекрестков.

Создаваемые классы: УмныйСветофор, Перекресток, ДатчикТранспортногоПотока, ИнженерАТС, ЦентрУправленияСветофорами.

Для классов реализовать следующие простые методы, используя для хранения данных списки ([]) Python:

(a) **УмныйСветофор:** изменить длительность фаз (красный/зеленый), перейти в аварийный режим, получить текущий режим работы.

(b) **Перекресток:** добавить светофор к перекрестку, удалить светофор, получить список всех светофоров перекрестка.

(c) **ДатчикТранспортногоПотока:** установить текущие данные о интенсивности движения, получить текущие показания, получить историю показаний.

(d) **ИнженерАТС:** класс, представляющий инженера автоматизированной транспортной системы, имеющий идентификатор, квалификацию, список закрепленных перекрестков, ФИО.

(e) **ЦентрУправленияСветофорами:** добавить новый перекресток в систему, установить умный светофор, нанять инженера, вывести информацию о перекрестках, светофорах, инженерах, удалить перекресток, уволить инженера, снять умный светофор.

28. **Описание ситуации:** Рассмотрим работу монорельсовой транспортной системы. Монорельс движется по эстакаде, состоящей из станций и перегонов. Составы имеют фиксированное количество вагонов. Операторы управляют движением составов, следят за соблюдением графика и безопасностью пассажиров.

    Создаваемые классы: СтанцияМонорельса, СоставМонорельса, ВагонМонорельса, ОператорСистемы, УправлениеМонорельсом.

    Для классов реализовать следующие простые методы, используя для хранения данных списки ([]) Python:

    (a) **СтанцияМонорельса:** принять состав, отправить состав, получить список составов на станции.

    (b) **СоставМонорельса:** добавить вагон в состав (при техническом обслуживании), удалить вагон, получить список вагонов.

    (c) **ВагонМонорельса:** зафиксировать текущий пробег, провести техническое обслуживание, получить историю обслуживаний.

    (d) **ОператорСистемы:** класс, представляющий оператора, имеющий идентификатор, смену, список закрепленных станций, ФИО.

    (e) **УправлениеМонорельсом:** добавить новую станцию, ввести состав в эксплуатацию, нанять оператора, вывести информацию о станциях, составах, операторах, закрыть станцию на ремонт, списать состав, уволить оператора.

29. **Описание ситуации:** Рассмотрим работу канатной дороги. Канатная дорога состоит из линий с опорами и кабинок, перемещающихся между станциями. Кабинки имеют ограниченную вместимость. Техники обслуживают механизмы и следят за безопасностью.

Создаваемые классы: ЛинияКанатнойДороги, Кабинка, СтанцияКанатнойДороги, Техник, УправлениеКанатнойДорогой.

Для классов реализовать следующие простые методы, используя для хранения данных списки ([]) Python:

(a) **ЛинияКанатнойДороги:** добавить кабинку на линию, снять кабинку, получить список кабинок на линии.

(b) **Кабинка:** запустить в движение, остановить для посадки/высадки, получить текущий статус (движется/стоит).

(c) **СтанцияКанатнойДороги:** принять кабинку, отправить кабинку, получить список кабинок на станции.

(d) **Техник:** класс, представляющий техника, имеющий идентификатор, квалификацию, список закрепленных линий, ФИО.

(e) **УправлениеКанатнойДорогой:** добавить новую линию, ввести кабинку в эксплуатацию, нанять техника, вывести информацию о линиях, кабинках, техниках, закрыть линию на обслуживание, списать кабинку, уволить техника.

30. **Описание ситуации:** Рассмотрим работу службы доставки с использованием дронов. Дроны осуществляют доставку небольших грузов между пунктами выдачи. Каждый дрон имеет грузоподъемность и дальность полета. Операторы управляют полетами дронов и обслуживают пункты выдачи.

    Создаваемые классы: ПунктВыдачи, Дрон, Груз, ОператорДронов, СлужбаДоставки.

    Для классов реализовать следующие простые методы, используя для хранения данных списки ([]) Python:

    (a) **ПунктВыдачи:** принять дрон с грузом, отправить дрон, получить список дронов в пункте.

    (b) **Дрон:** загрузить груз, выгрузить груз, начать полет, завершить полет, получить текущий статус (в полете/на земле).

    (c) **Груз:** зарегистрировать отправку, зарегистрировать доставку, получить историю перемещений.

    (d) **ОператорДронов:** класс, представляющий оператора, имеющий идентификатор, смену, список закрепленных пунктов выдачи, ФИО.

    (e) **СлужбаДоставки:** добавить новый пункт выдачи, ввести дрон в эксплуатацию, нанять оператора, вывести информацию о пунктах, дронах, операторах, закрыть пункт, списать дрон, уволить оператора.