Лабораторная работа №5

Время выполнения	@October 24, 2025 → November 14, 2025
Статус	Не начато

🚀 Задание "Космическая станция"

Необходимо создать систему классов для симуляции работы космической станции со следующими компонентами:

🎯 Цель работы

Разработать объектно-ориентированную систему управления космической станцией, применяя основные принципы ООП: инкапсуляцию, наследование, полиморфизм и абстракцию.

---

1. Базовый класс `CrewMember` (Член экипажа)

Атрибуты:

- `name` (str) — имя
- `rank` (str) — звание
- `health` (int, 0–100) — здоровье
- `energy` (int, 0–100) — энергия

Методы:

- `work()` — выполнить работу (**абстрактный метод**)
- `rest()` — отдохнуть
- `status_report()` — отчет о состоянии

---

2. Наследуемые классы экипажа

`Engineer` **(Инженер):**

- Специализация: ремонт систем
- Доп. атрибут: `repair_skill`

`Pilot` **(Пилот):**

- Специализация: управление кораблем
- Доп. атрибут: `flight_hours`

`Scientist` **(Ученый):**

- Специализация: исследования
- Доп. атрибут: `research_field`

---

3. Класс `Spacecraft` (Космический корабль)

Атрибуты:

- `name` (str) — название корабля
- `ship_type` (str) — тип корабля
- `crew_capacity` (int) — вместимость экипажа
- `current_crew` (list) — экипаж на борту
- `hull_integrity` (int, 0–100) — прочность корпуса

Методы:

- `add_crew_member(crew_member)` — добавить члена экипажа
- `remove_crew_member(crew_member)` — удалить члена экипажа
- `launch_mission(destination)` — запустить миссию

---

4. Класс `SpaceStation` (Космическая станция)

Атрибуты:

- `name` (str) — название станции
- `crew` (list) — экипаж станции

- `spacecraft_fleet` (list) — флот кораблей
- `resources` (dict) — ресурсы

Методы:

- `add_crew_member(crew_member)` — принять нового члена экипажа
- `assign_crew_to_ship(crew_members, spacecraft)` — назначить экипаж на корабль
- `daily_operations()` — ежедневные операции
- `generate_report()` — отчет о станции

📝 Реализуйте следующий сценарий выполнения:

1. Создайте объект космической станции класса `SpaceStation` и задайте ей название.
2. Создайте несколько членов экипажа разных специализаций (`Engineer`, `Pilot`, `Scientist`), задав их характеристики.
3. Добавьте созданных членов экипажа на станцию методом `add_crew_member()`.
4. Создайте космический корабль класса `Spacecraft`, указав его параметры (название, тип, вместимость, прочность корпуса).
5. Включите созданный корабль во флот станции.
6. Назначьте часть экипажа (например, пилота и инженера) на корабль методом `assign_crew_to_ship()`.
7. Выполните работу экипажем: вызовите метод `work()` у каждого члена экипажа и выведите их отчёты через `status_report()`.
8. Запустите миссию корабля, вызвав метод `launch_mission(destination)`, указав пункт назначения.
9. Организуйте работу учёного, оставшегося на станции: вызовите его метод `work()` и выведите отчёт.
10. Выполните ежедневные операции станции, вызвав метод `daily_operations()`.
11. Сформируйте и выведите итоговый отчёт о состоянии станции и её компонентов методом `generate_report()`.

Пример выполнения программы:

```
Лейтенант Иван прибыл на станцию Орбита-1.
Капитан Анна прибыл на станцию Орбита-1.
Сержант Олег прибыл на станцию Орбита-1.
Капитан Анна назначен на корабль Восток-7.
Лейтенант Иван назначен на корабль Восток-7.

=== Работа экипажа ===
Лейтенант Иван чинит системы (навык 75).
Лейтенант Иван | Здоровье: 90, Энергия: 65
Капитан Анна управляет кораблём (налёт 1200 часов).
Капитан Анна | Здоровье: 95, Энергия: 65
Сержант Олег проводит исследование в области: Астрофизика.
Сержант Олег | Здоровье: 88, Энергия: 60

=== Запуск миссии ===
Корабль Восток-7 отправляется на миссию в Луна!

=== Исследование ===
Сержант Олег проводит исследование в области: Астрофизика.
Сержант Олег | Здоровье: 88, Энергия: 50

=== Ежедневные операции ===
На станции Орбита-1 выполняются ежедневные операции.

=== Отчёт о станции ===
--- Отчёт о станции Орбита-1 ---
Экипаж станции:
  Лейтенант Иван | Здоровье: 90, Энергия: 65
  Капитан Анна | Здоровье: 95, Энергия: 65
  Сержант Олег | Здоровье: 88, Энергия: 50

Флот кораблей:
  Восток-7 (Шаттл), экипаж: 2

Ресурсы:
  еда: 90
  вода: 90
  кислород: 90
```

🎮 Бонусные задания

> ⚡ Для получения выше 80 баллов.
>
> Разумеется необходимо внятно объяснить вашу имплементацию для зачитывания.

🚀 Новые классы персонала

1. Класс `Medic` (Медик)

- Специализация: лечение членов экипажа
- Доп. атрибут: `medical_experience`
- Метод `heal(crew_member)` — восстанавливает здоровье другому члену экипажа

2. Класс `Security` (Охранник)

- Специализация: безопасность станции
- Доп. атрибут: `combat_skill`
- Метод `patrol()` — патрулирование станции

3. Класс `Chef` (Повар)

- Специализация: приготовление пищи
- Доп. атрибут: `cooking_skill`
- Метод `prepare_meal()` — готовит еду и восстанавливает энергию экипажа

🔧 Система повреждений и ремонта

4. Система поломок оборудования

- Добавить атрибут `equipment_status` в `SpaceStation`
- Случайные поломки систем (жизнеобеспечение, связь, навигация)
- Только инженеры могут их чинить

5. Критические повреждения кораблей

- Корабли имеют шанс получения повреждения в полете
- `hull_integrity` может падать ниже критического уровня
- Требуется экстренный ремонт или эвакуация

6. Система аварийных ситуаций

- Класс `Emergency` с типами аварий (пожар, разгерметизация, отказ систем)
- Разные специалисты справляются с разными типами аварий

💰 Экономическая система

7. Система торговли

- Класс `Trader` — торговец между станциями
- Покупка/продажа ресурсов с других станций
- Влияние цен на успех миссий

8. Бюджет и зарплаты

- У станции есть бюджет
- Экипаж получает зарплату в зависимости от ранга и результатов работы
- Недостаток средств влияет на моральный дух

9. Апгрейды и улучшения

- Возможность улучшать оборудование станции за ресурсы
- Улучшение кораблей (двигатели, защита, системы жизнеобеспечения)
- Повышение квалификации экипажа за опыт

🎯 Система миссий и квестов

10. Разнообразные типы миссий

- Исследовательские (требуют ученых)
- Спасательные (требуют медиков и пилотов)
- Грузовые (требуют опытных пилотов)
- Военные (требуют охранников)

11. Система случайных событий

- Встреча с астероидами
- Обнаружение новых планет
- Технические неполадки в полете

12. Цепочки связанных миссий

- Результат одной миссии влияет на доступность других
- Долгосрочные исследовательские проекты
- Построение репутации с различными фракциями

🧬 Система развития персонажей

13. Система опыта и уровней

- Члены экипажа получают опыт за выполнение работы
- Повышение уровня улучшает характеристики
- Изучение новых навыков

14. Система усталости и стресса

- Дополнительный атрибут `stress` (0-100)
- Длительная работа без отдыха увеличивает стресс
- Высокий стресс влияет на эффективность работы

15. Взаимоотношения в экипаже

- Атрибут `morale` для каждого члена экипажа
- Совместимость между членами экипажа
- Конфликты и дружба влияют на работу команды

🌌 Исследования и наука

16. Система научных открытий

- Ученые могут делать открытия, влияющие на всю станцию
- Исследование образцов с других планет
- Разработка новых технологий

17. База данных планет и систем

- Класс `Planet` с характеристиками (атмосфера, гравитация, ресурсы)
- Карта изученных систем
- Влияние условий планет на сложность миссий

🤖 Автоматизация и ИИ

18. Роботы и автоматизация

- Класс `Robot` — помощники для рутинных задач
- Роботы не устают, но могут ломаться
- Различные типы роботов (ремонтные, исследовательские, охранные)

19. ИИ система управления станцией

- Класс `StationAI` — искусственный интеллект станции
- Автоматическое планирование миссий
- Предупреждения о потенциальных проблемах
- Возможность "восстания" ИИ как особое событие

📊 Расширенная аналитика

20. Детальная статистика и логирование

- Система логов всех действий на станции
- Статистика эффективности каждого члена экипажа
- Анализ успешности различных типов миссий
- Прогнозирование потребности в ресурсах
- Графики изменения параметров станции во времени

🤔 Контрольные вопросы

1. Общие вопросы об ООП

1. Что такое объектно-ориентированное программирование?
2. Назовите четыре основных принципа ООП.
3. В чем заключается принцип инкапсуляции?
4. В чем заключается принцип наследования?
5. В чем заключается принцип полиморфизма?
6. Что такое абстракция в ООП?
7. Какие преимущества дает использование ООП?
8. В чем отличие объектно-ориентированного подхода от процедурного?

---

2. Классы и объекты

1. Что такое класс в Python?
2. Что такое объект (экземпляр класса)?
3. Как создать пустой класс в Python?
4. Как создать экземпляр класса?
5. Чем отличаются атрибуты класса от атрибутов экземпляра?
6. Как в Python организовать счетчик созданных объектов?
7. Как получить доступ к атрибуту класса через экземпляр?

---

3. Конструкторы и self

1. Какую роль выполняет метод `__init__`?
2. Зачем нужен параметр `self` в методах класса?
3. Можно ли вызывать методы экземпляра без `self`?
4. Что такое альтернативный конструктор в Python?
5. Как реализовать альтернативный конструктор с помощью `@classmethod`?
6. Как реализовать конструктор, который создает объект из строки?

---

4. Методы и поля

1. Чем отличаются методы экземпляра от методов класса?
2. Как объявить метод класса?
3. Чем методы класса отличаются от статических методов?
4. Как объявить статический метод?
5. Когда уместно использовать статические методы?
6. Что такое свойства (`@property`) в Python?
7. Как реализовать геттер и сеттер для свойства?
8. Чем отличается прямой доступ к атрибуту от использования свойства?

---

5. Инкапсуляция

1. Что такое инкапсуляция?
2. Чем отличаются публичные, защищенные и приватные атрибуты?
3. Как в Python обозначаются защищенные атрибуты?
4. Как в Python обозначаются приватные атрибуты?

5. Что такое name mangling?
6. Как получить доступ к приватному атрибуту через name mangling?
7. Почему прямой доступ к приватным атрибутам не рекомендуется?

---

6. Наследование

1. Что такое наследование?
2. Как в Python создать класс-наследник?
3. Как вызвать конструктор родительского класса?
4. Для чего используется функция `super()`?
5. Что такое переопределение метода?
6. Как реализуется множественное наследование в Python?
7. Что такое MRO (Method Resolution Order)?
8. Как посмотреть порядок разрешения методов (MRO) у класса?

---

7. Полиморфизм

1. Что такое полиморфизм?
2. Что такое duck typing в Python?
3. В чем отличие `isinstance()` от `type()`?
4. Когда следует использовать `isinstance()`, а когда `type()`?

---

8. Абстракция

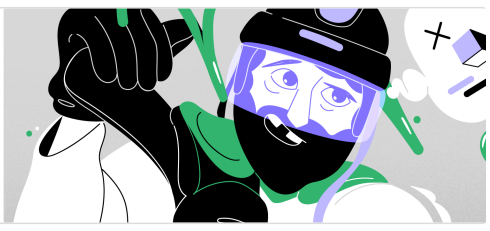
1. Что такое абстрактный класс?
2. Как объявить абстрактный метод в Python?

📖 Дополнительные ресурсы

Объектно-ориентированное программирование в размышлениях и опытах хоккеиста Степана Осечкина

В программировании есть несколько основных парадигм — подходов, которые определяют, как программисты структурируют и организуют свой код. Например, процедурное программирование — самый простой и...

https://habr.com/ru/companies/yandex_praktikum/articles/749180/



cmp.phys.msu.su

https://cmp.phys.msu.su/sites/default/files/%D0%9E%D0%9E%D0%9F_%D0%BD%D0%B0_Python_%D0%A3%D1%87%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5%20%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5_var7.pdf

---

W3Schools.com

W3Schools offers free online tutorials, references and exercises in all the major languages of the web. Covering popular subjects like HTML, CSS, JavaScript, Python, SQL, Java, and many, many more.

https://www.w3schools.com/python/python_classes.asp

Object-Oriented Programming (OOP) in Python – Real Python

In this tutorial, you'll learn all about object-oriented programming (OOP) in Python. You'll learn the basics of the OOP paradigm and cover concepts like classes and inheritance. You'll also see how to instantiate an object from a class.

https://realpython.com/python3-object-oriented-programming/