Отчет по лабораторным работам

Дата: 05-12-2025
Семестр: 2 курс, 1 семестр
Группа: ПИН-Б-О-24-2
Дисциплина: Технологии программирования
Студент: Осипов Александр Сергеевич

Цель работы

Изучить ключевые принципы и механизмы объектно-ориентированного программирования: 1. **Инкапсуляция**: Реализация сокрытия данных и доступа через свойства. 2. **Наследование и Абстракция**: Построение иерархий классов и определение общих интерфейсов. 3. **Полиморфизм и Магические методы**: Работа с объектами разных типов через единый интерфейс и перегрузка операторов. 4. **Композиция и Агрегация**: Создание сложных объектных структур и управление связями между ними.

Теоретическая часть

Лабораторная работа №1: Инкапсуляция

Инкапсуляция - это один из ключевых принципов ООП, который заключается в сокрытии внутренней реализации объекта и предоставлении строго определенного интерфейса для взаимодействия с ним. В Python инкапсуляция реализуется через:

1. **Приватные атрибуты** - атрибуты, начинающиеся с двойного подчеркивания (`__attribute`), которые не могут быть напрямую доступны извне класса.
2. **Свойства (properties)** - механизм доступа к приватным атрибутам через методы-геттеры и сеттеры с использованием декораторов `@property` и `@<attribute>.setter`.

3. **Валидация данных** - проверка корректности данных при их установке через сеттеры.

Лабораторная работа №2: Наследование и Абстракция

Наследование - это механизм ООП, позволяющий создавать новый класс на основе существующего, наследуя его атрибуты и методы. Это обеспечивает повторное использование кода и построение иерархий классов.

Абстракция - это принцип ООП, позволяющий определять общий интерфейс для группы связанных классов через абстрактные классы и методы. В Python абстракция реализуется через модуль `abc` (Abstract Base Classes).

Паттерн Factory Method - это порождающий паттерн проектирования, который предоставляет интерфейс для создания объектов, позволяя подклассам решать, какой класс инстанцировать.

Лабораторная работа №3: Полиморфизм и Магические методы

Полиморфизм - это способность объектов разных классов обрабатываться через единый интерфейс. В Python полиморфизм реализуется через общие методы в базовых классах и их переопределение в дочерних классах.

Магические методы (dunder methods) - это специальные методы в Python, которые начинаются и заканчиваются двойным подчеркиванием (например, `__init__`, `__str__`, `__eq__`). Они позволяют переопределять поведение операторов и встроенных функций.

Сериализация - процесс преобразования объектов в формат, пригодный для хранения или передачи (например, JSON).

Лабораторная работа №4: Композиция и Агрегация

Композиция - это отношение "часть-целое", где часть не может существовать без целого. При удалении целого удаляются и все его части.

Агрегация - это отношение "часть-целое", где часть может существовать независимо от целого. При удалении целого части остаются.

Валидация данных - процесс проверки корректности данных перед их использованием. Реализуется через кастомные исключения и проверки в методах.

Практическая часть

Выполненные задачи

Лабораторная работа №1

- Создан класс `Employee` с приватными атрибутами.
- Реализованы свойства (геттеры и сеттеры) с валидацией данных.
- Обработаны исключения при некорректном вводе данных.
- Создана демонстрационная программа.

Лабораторная работа №2

- Создан абстрактный класс `AbstractEmployee`.
- Реализованы подклассы `Manager`, `Developer`, `Salesperson`.
- Внедрен паттерн Factory Method (`EmployeeFactory`).
- Переопределены методы расчета зарплаты для каждого типа сотрудников.

Лабораторная работа №3

- Реализован класс `Department` для управления коллекцией сотрудников.
- Перегружены операторы сравнения (`__eq__`, `__lt__`) и арифметические операторы (`__add__`).
- Реализованы методы для итерации (`__iter__`) и доступа по индексу (`__getitem__`).
- Добавлена сериализация (`to_dict`, `from_dict`) и сохранение в файл.

Лабораторная работа №4

- Реализован класс `Project` (композиция команды).
- Реализован класс `Company` (агрегация отделов и проектов).
- Созданы кастомные исключения и валидация.
- Реализован экспорт отчетов в CSV.
- Добавлены функции анализа и планирования.

Ключевые фрагменты кода

Лабораторная работа №1

```
class Employee:
    @property
    def id(self) -> int:
        return self.__id

    @id.setter
    def id(self, value: int):
        if not isinstance(value, int) or value <= 0:
            raise ValueError("ID должен быть положительным целым числом")
        self.__id = value
```

Лабораторная работа №2

```
class Manager(Employee):
    def calculate_salary(self) -> float:
        return self.base_salary + self.__bonus

class EmployeeFactory:
    @staticmethod
    def create_employee(emp_type: str, **kwargs) -> AbstractEmployee:
        if emp_type.lower() == "manager":
            return Manager(**kwargs)
        # ...
```

Лабораторная работа №3

```
def __add__(self, other) -> float:
    """Сложение возвращает сумму зарплат."""
    if not isinstance(other, AbstractEmployee):
        return NotImplemented
    return self.calculate_salary() + other.calculate_salary()

def __iter__(self):
    """Итератор по сотрудникам: for emp in dept"""
    return iter(self.__employees)
```

Лабораторная работа №4

```
class Project:
    def __init__(self, ...):
        self.__team: list[AbstractEmployee] = []  # Композиция

class Company:
    def __init__(self, ...):
        self.__departments: list[Department] = []  # Агрегация
        self.__projects: list[Project] = []       # Агрегация

    def assign_employee_to_project(self, employee_id: int, project_id: int) ->
bool:
        # Логика назначения с проверками
```

Результаты выполнения

Пример работы программ

Лабораторная работа №1

```
Создан: Сотрудник [id: 1, имя: Иван Петров, отдел: Разработка, базовая зарплата:
50000.0]
Попытка установки невалидных значений:
Корректно обработана ошибка ID: ID должен быть положительным числом
```

Лабораторная работа №2

```
Создан менеджер: Мария Сидорова
Создан разработчик: Алексей Иванов
Manager: 2 чел., средняя зарплата: 81500.00
Developer: 2 чел., средняя зарплата: 83750.00
```

Лабораторная работа №3

```
Общая зарплата всех сотрудников отдела: 284500.00
developer1 + developer2 = 167500.00
manager in dept: True
Сортировка по зарплате (убывание):
```

```
  Иван Сидоров: 100000.00
  Анна Петрова: 72000.00
```

Лабораторная работа №4

```
 Создана компания: TechInnovations
Компания сохранена в JSON: data/json/company_data.json
Статистика по отделам:
  Development: 2 сотрудников, зарплата: 17200.00
Анализ проектов:
  Всего проектов: 2
  Общий бюджет: 17200.00
```

Тестирование

- **Лабораторная работа №1**: Модульные тесты пройдены, валидация корректно отрабатывает исключения.
- **Лабораторная работа №2**: Интеграционные тесты подтверждают корректность наследования и полиморфизма. Фабричный метод работает штатно.
- **Лабораторная работа №3**: Магические методы ведут себя ожидаемо (`+`, `==`, `<`, `in`). Сериализация/десериализация сохраняет состояние объектов.
- **Лабораторная работа №4**: Композиция и агрегация функционируют верно. Экспорт в CSV и анализ данных выдают корректные результаты.

Выводы

Общие выводы по всем лабораторным работам

1. **Инкапсуляция и безопасность данных**: Использование приватных атрибутов и свойств позволяет создать надежный интерфейс класса, защищающий данные от некорректного использования и упрощающий поддержку кода.
2. **Гибкость через наследование и абстракцию**: Иерархии классов и абстрактные базовые классы позволяют писать расширяемый код, где добавление новых сущностей не требует переписывания существующей логики.

3. **Полиморфизм и удобство использования**: Единые интерфейсы и перегрузка операторов (магические методы) делают код более читаемым, интуитивным и близким к естественному языку, позволяя работать с разными объектами одинаково.
4. **Архитектура сложных систем**: Композиция и агрегация являются фундаментальными инструментами для моделирования связей реального мира. Правильное их использование (сильная vs слабая связь) критично для жизненного цикла объектов.
5. **Персистентность и анализ**: Реализация сериализации (JSON) и экспорта данных (CSV) обеспечивает сохранение состояния системы и возможность интеграции с внешними инструментами аналитики.

Ответы на контрольные вопросы

Лабораторная работа №1

1. **Что такое инкапсуляция?** Принцип сокрытия внутренней реализации и предоставления доступа через интерфейс.
2. **Как создать приватный атрибут?** Использовать префикс `__` (например, `__name`).
3. **Зачем нужна валидация в сеттерах?** Для обеспечения целостности данных и предотвращения ошибок.

Лабораторная работа №2

1. **Что такое абстрактный класс?** Класс, который нельзя инстанцировать, содержащий абстрактные методы для реализации в потомках.
2. **Factory Method?** Паттерн, делегирующий создание объектов подклассам/фабрике.
3. **Наследование vs Композиция?** "Является" (is-a) против "Имеет" (has-a).

Лабораторная работа №3

1. **Полиморфизм?** Способность объектов разных типов обрабатываться одним интерфейсом.
2. **Магические методы?** Методы вроде `__init__`, `__add__`, переопределяющие поведение операторов.
3. **Сериализация?** Перевод объекта в байты/строку (JSON) для хранения.

Лабораторная работа №4

1. **Композиция vs Агрегация?** Композиция - жесткая связь (умирают вместе), Агрегация - мягкая (части живут отдельно).
2. **Кастомные исключения?** Улучшают читаемость и обработку специфичных ошибок.
3. **Как работает сериализация сложных структур?** Рекурсивный обход всех вложенных объектов и связей.

Структура проекта

```
.
├── lab01/              # Инкапсуляция
│   ├── project/
│   │   └── employee.py
│   ├── report/
│   │   ├── report.md
│   │   └── uml_diagram.puml
│   └── task.md
├── lab02/              # Наследование
│   ├── project/
│   │   └── main.py
│   ├── report/
│   │   └── report.md
│   └── task.md
├── lab03/              # Полиморфизм
│   ├── project/
│   │   └── main.py
│   ├── report/
│   │   └── report.md
│   └── task.md
├── lab04/              # Композиция и Агрегация
│   ├── employee_management_system/
│   │   ├── src/
│   │   ├── examples/
│   │   └── data/
│   ├── report/
│   │   └── report.md
│   └── task.md
└── ОТЧЕТ.md            # Итоговый отчет
```