Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6
дисциплины
«Объектно-ориентированное программирование»
Вариант 15

Выполнил:
Степанов Леонид Викторович
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Богданов Сергей Сергеевич,
Ассистент департамента цифровых, робототехнических систем и электроники

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Классы данных в Python

Цель: приобретение навыков по работе с классами данных при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

Пример. Для примера 2 лабораторной работы 9 добавьте возможность работы с классами данных (рис. 1), а также сохранения и чтения данных в формат XML (рис.2).

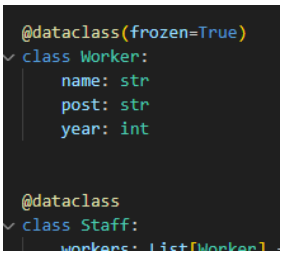

Рисунок 1 – Датакласс

```
def load(self, filename):
    with open(filename, "r", encoding="utf8") as fin:
        xml = fin.read()
    parser = ET.XMLParser(encoding="utf8")
    tree = ET.fromstring(xml, parser=parser)
    self.workers = []
    for worker_element in tree:
        name, post, year = None, None, None
        for element in worker_element:
            if element.tag == "name":
                name = element.text
            elif element.tag == "post":
                post = element.text
            elif element.tag == "year":
                year = int(element.text)
            if name is not None and post is not None and year is not None:
                self.workers.append(Worker(name=name, post=post, year=year))

def save(self, filename):
    root = ET.Element("workers")
    for worker in self.workers:
        worker_element = ET.Element("worker")
        name_element = ET.SubElement(worker_element, "name")
        name_element.text = worker.name
        post_element = ET.SubElement(worker_element, "post")
        post_element.text = worker.post
        year_element = ET.SubElement(worker_element, "year")
        year_element.text = str(worker.year)
        root.append(worker_element)
    tree = ET.ElementTree(root)
    with open(filename, "wb") as fout:
        tree.write(fout, encoding="utf8", xml_declaration=True)
```

Рисунок 2 – Чтение и сохранение в формат XML

Индивидуальное задание: выполнить индивидуальное задание лабораторной работы 4.5, использовав классы данных, а также загрузку и сохранение данных в формат XML.

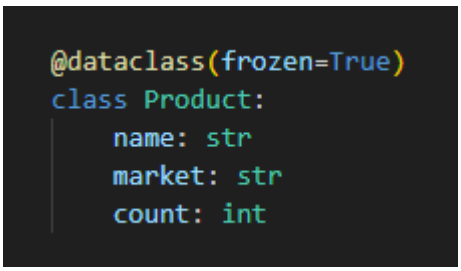

Рисунок 3 – Датакласс

```
def load(self, filename: str) -> None:
    with open(filename, "r", encoding="utf8") as fin:
        xml = fin.read()
    parser = ET.XMLParser(encoding="utf8")
    tree = ET.fromstring(xml, parser=parser)
    self.products = []
    for product_element in tree:
        name, market, count = None, None, None
        for element in product_element:
            if element.tag == "name":
                name = element.text
            elif element.tag == "market":
                market = element.text
            elif element.tag == "count":
                count = int(element.text)
            if name is not None and market is not None and count is not None:
                self.products.append(Product(name=name, market=market, count=count))

def save(self, filename: str) -> None:
    root = ET.Element("products")
    for product in self.products:
        product_element = ET.Element("product")
        name_element = ET.SubElement(product_element, "name")
        name_element.text = product.name
        market_element = ET.SubElement(product_element, "market")
        market_element.text = product.market
        count_element = ET.SubElement(product_element, "count")
        count_element.text = str(product.count)
        root.append(product_element)
    tree = ET.ElementTree(root)
    with open(filename, "wb") as fout:
        tree.write(fout, encoding="utf8", xml_declaration=True)
```

Рисунок 4 – Чтение и сохранение в формат XML

Ссылка на github: https://github.com/FiaLDI/lab4.6

Контрольные вопросы

1. Как создать класс данных в языке Python?

В Python для создания класса данных используется модуль dataclasses, который позволяет автоматически генерировать специальные методы, такие как __init__() и __repr__(). Чтобы создать класс данных, необходимо использовать декоратор @dataclass

2. Какие методы по умолчанию реализует класс данных?

Классы данных автоматически реализуют несколько специальных методов, включая:

1) _init__(): для инициализации экземпляров класса,
2) __repr__(): для представления экземпляров класса в виде строки,
3) __eq__(): для сравнения экземпляров класса на равенство,
4) __hash__(): если класс данных является неизменяемым, этот метод также будет сгенерирован,

Эти методы позволяют избежать написания шаблонного кода и делают работу с классами данных более удобной.

3. Как создать неизменяемый класс данных?

Чтобы создать неизменяемый класс данных, нужно использовать параметр frozen=True в декораторе @dataclass. Это сделает все атрибуты класса неизменяемыми после их инициализации. Вот пример: @dataclass(frozen=True)

Вывод: в ходе выполнения работы были приобретены навыки по работе с классами данных при написании программ с помощью языка программирования Python версии 3.x.