

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 №4.6
дисциплины «Объектно-ориентированное программирование»
Вариант 2

Выполнил:
Иващенко Олег Андреевич
3 курс, группа ИВТ-б-о-22-1,
09.03.02 «Информационные и
вычислительные машины»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем»

(подпись)

Руководитель практики:
Воронкин Роман Александрович,
доцент департамента цифровых,
робототехнических систем и
электроники

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: «Классы данных»

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

Порядок выполнения работы:

Индивидуальное задание. Выполнить индивидуальное задание лабораторной работы 4.5, используя классы данных, а также загрузку и сохранение данных в формат XML.

Таблица 1 – Листинг программы

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Выполнить индивидуальное задание лабораторной работы 4.5, используя
классы данных, а также загрузку и сохранение данных в формат XML.
"""

import os
import argparse
import xml.etree.ElementTree as ET

class FileNode:
    """Класс для представления узла дерева (файла или папки)."""

    def __init__(self, name, is_dir):
        """Инициализация узла."""
        self.name = name
        self.is_dir = is_dir
        self.children = []

    def build_tree(path, level, max_level, show_hidden):
        """
        Рекурсивно строит дерево каталогов.

        :param path: Путь к текущему каталогу.
        :param level: Текущий уровень вложенности.
        :param max_level: Максимальный уровень вложенности.
        :param show_hidden: Показывать ли скрытые файлы.
        :return: Узел дерева.
        """
        if level > max_level:
            return None
```

```

name = os.path.basename(path)
is_dir = os.path.isdir(path)
node = FileNode(name, is_dir)

if is_dir:
    for item in os.listdir(path):
        if not show_hidden and item.startswith('.'):
            continue
        child_path = os.path.join(path, item)
        child_node = build_tree(child_path, level + 1, max_level, show_hidden)
        if child_node:
            node.children.append(child_node)

return node

def print_tree(node, level=0):
    """
    Выводит дерево каталогов на экран.

    :param node: Корневой узел дерева.
    :param level: Текущий уровень вложенности.
    """
    if not node:
        return

    indent = " " * level
    print(f"{indent}/{node.name}" if node.is_dir else f"{indent}{node.name}")
    for child in node.children:
        print_tree(child, level + 1)

def save_to_xml(node, parent_element=None):
    """
    Сохраняет дерево каталогов в XML.

    :param node: Корневой узел дерева.
    :param parent_element: Родительский XML-элемент.
    :return: XML-элемент.
    """
    if parent_element is None:
        element = ET.Element("directory", {"name": node.name})
    else:
        element = ET.SubElement(
            parent_element,
            "directory" if node.is_dir else "file",
            {"name": node.name}
        )

    for child in node.children:
        save_to_xml(child, element)

```

```

return element

def load_from_xml(element):
    """
    Загружает дерево каталогов из XML.

    :param element: XML-элемент.
    :return: Узел дерева.
    """
    name = element.attrib["name"]
    is_dir = element.tag == "directory"
    node = FileNode(name, is_dir)

    for child_element in element:
        child_node = load_from_xml(child_element)
        node.children.append(child_node)

    return node

def main():
    """Основная функция программы."""
    parser = argparse.ArgumentParser(description="Утилита для отображения дерева каталогов.")
    parser.add_argument("directory", nargs="?", default=".", help="Путь к каталогу")
    parser.add_argument("-l", "--level", type=int, default=100,
                        help="Максимальная глубина отображения")
    parser.add_argument("-a", "--all", action="store_true",
                        help="Показывать скрытые файлы")
    parser.add_argument("--save", type=str, help="Сохранить дерево в XML файл")
    parser.add_argument("--load", type=str, help="Загрузить дерево из XML файла")

    args = parser.parse_args()

    if args.load:
        tree_xml = ET.parse(args.load)
        root = load_from_xml(tree_xml.getroot())
        print("Дерево загружено из XML:")
        print_tree(root)
        return

    path = os.path.abspath(args.directory)
    if not os.path.exists(path):
        print("Ошибка: каталог не существует.")
        return

    if not os.path.isdir(path):
        print("Ошибка: указанный путь не является каталогом.")
        return

    root = build_tree(path, 0, args.level, args.all)

```

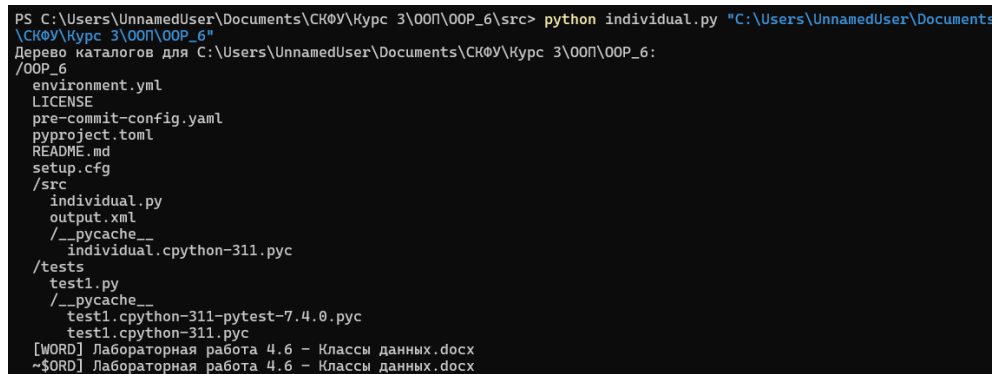
```

print(f"Дерево каталогов для {path}:")
print_tree(root)

if args.save:
    root_xml = save_to_xml(root)
    tree_xml = ET.ElementTree(root_xml)
    tree_xml.write(args.save, encoding="utf-8", xml_declaration=True)
    print(f"Дерево сохранено в {args.save}")

if __name__ == "__main__":
    main()

```



```

PS C:\Users\UnnamedUser\Documents\СКФУ\Курс 3\ООП\ООП_6\src> python individual.py "C:\Users\UnnamedUser\Documents\СКФУ\Курс 3\ООП\ООП_6"
Дерево каталогов для C:\Users\UnnamedUser\Documents\СКФУ\Курс 3\ООП\ООП_6:
/ООП_6
  environment.yml
  LICENSE
  pre-commit-config.yaml
  pyproject.toml
  README.md
  setup.cfg
  /src
    individual.py
    output.xml
    /__pycache__
      individual.cpython-311.pyc
  /tests
    test1.py
    /__pycache__
      test1.cpython-311-pytest-7.4.0.pyc
      test1.cpython-311.pyc
[WORD] Лабораторная работа 4.6 - Классы данных.docx
~$ORD Лабораторная работа 4.6 - Классы данных.docx

```

Рисунок 1 – Вывод программы



```

PS C:\Users\UnnamedUser\Documents\СКФУ\Курс 3\ООП\ООП_6\src> python individual.py "C:\Users\UnnamedUser\Documents\СКФУ\Курс 3\ООП\ООП_6" --save
output.xml
Дерево каталогов для C:\Users\UnnamedUser\Documents\СКФУ\Курс 3\ООП\ООП_6:
/ООП_6
  environment.yml
  LICENSE
  pre-commit-config.yaml
  pyproject.toml
  README.md
  setup.cfg
  /src
    individual.py
    /__pycache__
      individual.cpython-311.pyc
  /tests
    test1.py
    /__pycache__
      test1.cpython-311-pytest-7.4.0.pyc
      test1.cpython-311.pyc
[WORD] Лабораторная работа 4.6 - Классы данных.docx
~$ORD Лабораторная работа 4.6 - Классы данных.docx
~\RL6427.tmp
Дерево сохранено в output.xml

```

Рисунок 2 – Сохранение дерева каталогов в файл

```
This XML file does not appear to have any style information associated with it. The document tree is shown below.

<?xml version="1.0" encoding="utf-8" ?>
<directory name="00P_6">
  <file name="environment.yml"/>
  <file name="LICENSE"/>
  <file name="pre-commit-config.yaml"/>
  <file name="pyproject.toml"/>
  <file name="README.md"/>
  <file name="setup.cfg"/>
  <directory name="src">
    <file name="individual.py"/>
    <directory name="__pycache__">
      <file name="individual.cpython-311.pyc"/>
    </directory>
  </directory>
  <directory name="tests">
    <file name="test1.py"/>
    <directory name="__pycache__">
      <file name="test1.cpython-311-pytest-7.4.0.pyc"/>
      <file name="test1.cpython-311.pyc"/>
    </directory>
  </directory>
  <file name="[WORD] Лабораторная работа 4.6 - Классы данных.docx"/>
  <file name="~$ORD] Лабораторная работа 4.6 - Классы данных.docx"/>
  <file name="~WRL0427.tmp"/>
</directory>
```

Рисунок 3 – Содержание сохранённого файла

```
PS C:\Users\UnnamedUser\Documents\КФУ\Курс 3\00П\00P_6\src> python individual.py --load output.xml
Дерево загружено из XML:
/00P_6
  environment.yml
  LICENSE
  pre-commit-config.yaml
  pyproject.toml
  README.md
  setup.cfg
  /src
    individual.py
    /__pycache__
      individual.cpython-311.pyc
  /tests
    test1.py
    /__pycache__
      test1.cpython-311-pytest-7.4.0.pyc
      test1.cpython-311.pyc
  [WORD] Лабораторная работа 4.6 - Классы данных.docx
  ~$ORD] Лабораторная работа 4.6 - Классы данных.docx
  ~WRL0427.tmp
```

Рисунок 4 – Вывод сохранённого файла в консоль

Таблица 2 – Листинг теста

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import os
import xml.etree.ElementTree as ET
from src.individual import FileNode, build_tree, print_tree, save_to_xml, load_from_xml

def test_build_tree():
    test_dir = "test_dir"
    os.makedirs(test_dir, exist_ok=True)
    with open(os.path.join(test_dir, "file1.txt"), "w") as f:
        f.write("test")
    os.makedirs(os.path.join(test_dir, "subdir"), exist_ok=True)

    root = build_tree(test_dir, 0, 10, False)
```

```
assert root.name == "test_dir"
assert root.is_dir is True
assert len(root.children) == 2
assert root.children[0].name == "file1.txt"
assert root.children[0].is_dir is False
assert root.children[1].name == "subdir"
assert root.children[1].is_dir is True

os.remove(os.path.join(test_dir, "file1.txt"))
os.rmdir(os.path.join(test_dir, "subdir"))
os.rmdir(test_dir)
```

```
def test_save_and_load_xml():
    root = FileNode("test_dir", True)
    file_node = FileNode("file1.txt", False)
    subdir_node = FileNode("subdir", True)
    root.children.append(file_node)
    root.children.append(subdir_node)

    xml_file = "test.xml"
    root_xml = save_to_xml(root)
    tree_xml = ET.ElementTree(root_xml)
    tree_xml.write(xml_file, encoding="utf-8", xml_declaration=True)

    tree_xml = ET.parse(xml_file)
    loaded_root = load_from_xml(tree_xml.getroot())

    assert loaded_root.name == "test_dir"
    assert loaded_root.is_dir is True
    assert len(loaded_root.children) == 2
    assert loaded_root.children[0].name == "file1.txt"
    assert loaded_root.children[0].is_dir is False
    assert loaded_root.children[1].name == "subdir"
    assert loaded_root.children[1].is_dir is True

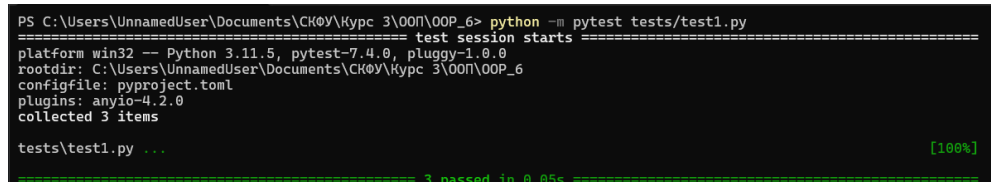
    os.remove(xml_file)
```

```
def test_print_tree(capsys):
    root = FileNode("test_dir", True)
    file_node = FileNode("file1.txt", False)
    subdir_node = FileNode("subdir", True)
    root.children.append(file_node)
    root.children.append(subdir_node)

    print_tree(root)

    captured = capsys.readouterr()
    expected_output = "/test_dir\n file1.txt\n /subdir\n"
    assert captured.out == expected_output
```

```
if __name__ == "__main__":
    test_build_tree()
    test_save_and_load_xml()
    test_print_tree()
```



```
PS C:\Users\UnnamedUser\Documents\СКФУ\Курс 3\ООП\ООП_6> python -m pytest tests/test1.py
===== test session starts =====
platform win32 -- Python 3.11.5, pytest-7.4.0, pluggy-1.0.0
rootdir: C:\Users\UnnamedUser\Documents\СКФУ\Курс 3\ООП\ООП_6
configfile: pyproject.toml
plugins: anyio-4.2.0
collected 3 items

tests\test1.py ... [100%]

===== 3 passed in 0.05s =====
```

Рисунок 5 – Вывод теста программы

Ответы на контрольные вопросы:

1. Как создать класс данных в языке Python?

Класс данных создается с помощью декоратора `@dataclass` из модуля `dataclasses`. Этот декоратор автоматически добавляет стандартные методы, такие как `__init__`, `__repr__` и `__eq__`, на основе объявленных атрибутов.

2. Какие методы по умолчанию реализует класс данных?

Класс данных автоматически реализует следующие методы:

- `__init__` - конструктор для инициализации атрибутов.
- `__repr__` - метод для строкового представления объекта (удобный вывод).
- `__eq__` - метод для сравнения объектов по значениям атрибутов.

3. Как создать неизменяемый класс данных?

Чтобы сделать класс данных неизменяемым, используется параметр `frozen=True` в декораторе `@dataclass`, запрещающий изменение атрибутов после создания объекта.

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

Ссылка на репозиторий GitHub:

[IUnnamedUserI/OOP_6: Объектно-ориентированное программирование.](#)
[Лабораторная работа №6](#)