

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»

Выполнил:
Кожуховский Виктор Андреевич
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:
Воронкин Роман Александрович

(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:

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.
3. Выполнил клонирование созданного репозитория.
4. Дополнил файл .gitignore необходимыми правилами для работы с IDE.
5. Организовал свой репозиторий в соответствии с моделью ветвления git-flow.
6. Создал проект в папке репозитория.
7. Проработал примеры лабораторной работы.
8. Выполнил индивидуальное задание для варианта 14.

Задание 1

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

```
import os
import argparse
import stat
import xml.etree.ElementTree as ET
from dataclasses import dataclass, field
from typing import List, Union

@dataclass
class File:
    name: str
    size: int

@dataclass
class Directory:
    path: str
    files: List[File] = field(default_factory=list)
    subdirs: List["Directory"] = field(default_factory=list)

def is_hidden(filepath: str) -> bool:
    if os.path.basename(filepath).startswith("."):
        return True
    try:
        return bool(os.stat(filepath).st_file_attributes & stat.FILE_ATTRIBUTE_HIDDEN)
    except AttributeError:
        return False

def list_files(
```

```

startpath: str, depth: int = -1, level: int = 0, show_hidden: bool = False
) -> Directory:
    if depth == 0:
        return Directory(path=startpath)

directory = Directory(path=startpath)

for element in os.listdir(startpath):
    path = os.path.join(startpath, element)
    if (element.startswith(".") or is_hidden(path)) and not show_hidden:
        continue
    if os.path.isdir(path):
        sub_directory = list_files(path, depth - 1, level + 1, show_hidden)
        directory.subdirs.append(sub_directory)
    else:
        size = os.path.getsize(path)
        directory.files.append(File(name=element, size=size))

return directory

def save_to_xml(directory: Directory, xml_file: Union[str, ET.Element]) -> ET.Element:
    if isinstance(xml_file, str):
        root = ET.Element("directory", path=directory.path)
    else:
        root = xml_file
        root.set("path", directory.path)

    for file in directory.files:
        ET.SubElement(root, "file", name=file.name, size=str(file.size))

    for subdir in directory.subdirs:
        subdir_element = ET.SubElement(root, "subdirectory")
        save_to_xml(subdir, subdir_element)

    if isinstance(xml_file, str):
        tree = ET.ElementTree(root)
        tree.write(xml_file, encoding="utf-8", xml_declaration=True)

    return root

def load_from_xml(xml_source: Union[str, ET.Element]) -> Directory:
    if isinstance(xml_source, str):
        tree = ET.parse(xml_source)
        root = tree.getroot()
    else:
        root = xml_source

    directory = Directory(path=root.get("path", ""))

    for file_element in root.findall("file"):
        name = file_element.get("name", "")
        size = int(file_element.get("size", 0))
        directory.files.append(File(name=name, size=size))

    for subdir_element in root.findall("subdirectory"):
        subdir = load_from_xml(subdir_element)
        directory.subdirs.append(subdir)

    return directory

def print_directory(directory: Directory, level: int = 0) -> None:
    indent = " " * 4 * level
    for file in directory.files:
        print(f'{indent} └── {file.name} ( {file.size} bytes)')
    for subdir in directory.subdirs:
        print(f'{indent} └── {os.path.basename(subdir.path)}/')
        print_directory(subdir, level + 1)

def main() -> None:
    parser = argparse.ArgumentParser(description="Display directory tree")
    parser.add_argument("path", nargs="?", default="", help="Directory path")
    parser.add_argument("-d", "--depth", type=int, default=-1, help="Depth of the tree")
    parser.add_argument("--hidden", action="store_true", help="Show hidden files")
    parser.add_argument("--save", type=str, help="Save directory structure to XML file")
    parser.add_argument(

```

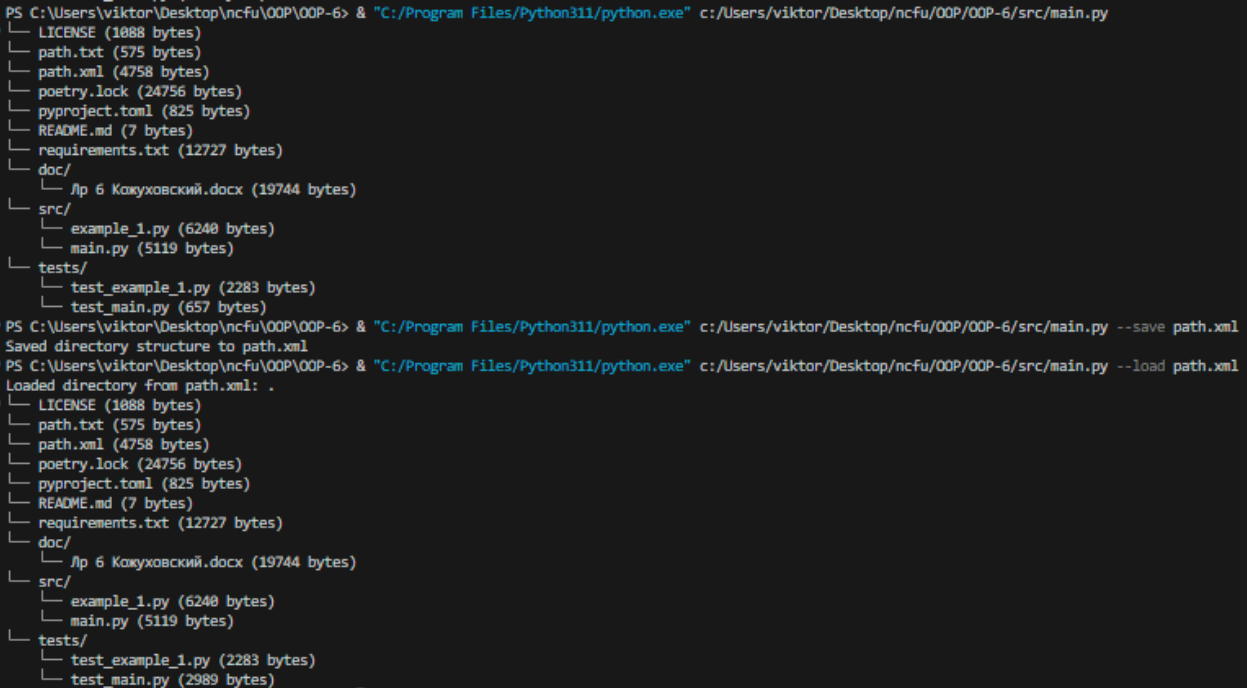
```

        "--load", type=str, help="Load directory structure from XML file"
    )
    args = parser.parse_args()

    if args.load:
        directory = load_from_xml(args.load)
        print(f"Loaded directory from {args.load}: {directory.path}")
        print_directory(directory)
    else:
        directory = list_files(args.path, args.depth, show_hidden=args.hidden)
        if args.save:
            save_to_xml(directory, args.save)
            print(f"Saved directory structure to {args.save}")
        else:
            print_directory(directory)

    if __name__ == "__main__":
        main()

```



```

PS C:\Users\viktor\Desktop\ncfu\OOP\OOP-6> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/ncfu/OOP/OOP-6/src/main.py
├─ LICENSE (1088 bytes)
├─ path.txt (575 bytes)
├─ path.xml (4758 bytes)
├─ poetry.lock (24756 bytes)
├─ pyproject.toml (825 bytes)
├─ README.md (7 bytes)
├─ requirements.txt (12727 bytes)
├─ doc/
│   └─ /р 6 Кажухавский.docx (19744 bytes)
├─ src/
│   ├── example_1.py (6240 bytes)
│   └─ main.py (5119 bytes)
├─ tests/
│   ├── test_example_1.py (2283 bytes)
│   └─ test_main.py (657 bytes)
PS C:\Users\viktor\Desktop\ncfu\OOP\OOP-6> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/ncfu/OOP/OOP-6/src/main.py --save path.xml
Saved directory structure to path.xml
PS C:\Users\viktor\Desktop\ncfu\OOP\OOP-6> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/ncfu/OOP/OOP-6/src/main.py --load path.xml
Loaded directory from path.xml: .
├─ LICENSE (1088 bytes)
├─ path.txt (575 bytes)
├─ path.xml (4758 bytes)
├─ poetry.lock (24756 bytes)
├─ pyproject.toml (825 bytes)
├─ README.md (7 bytes)
├─ requirements.txt (12727 bytes)
├─ doc/
│   └─ /р 6 Кажухавский.docx (19744 bytes)
├─ src/
│   ├── example_1.py (6240 bytes)
│   └─ main.py (5119 bytes)
├─ tests/
│   ├── test_example_1.py (2283 bytes)
│   └─ test_main.py (2989 bytes)

```

Рисунок 1. Код решения индивидуального задания 1 и его выполнение

9. Зафиксировал сделанные изменения в репозитории.

10. Выполнил слияние ветки для разработки с веткой master/main.

11. Отправил сделанные изменения на сервер GitHub.

Ссылка: <https://github.com/Viktorkozh/OOP-6>

Контрольные вопросы:

1. Как создать класс данных в языке Python?

```
from dataclasses import dataclass
```

```
@dataclass
```

```
class DataClassCard:
```

```
rank: str
```

```
suit: str
```

2. Какие методы по умолчанию реализует класс данных?

По умолчанию, классы данных реализует метод `__repr__()`, чтобы предоставить хорошее строковое представление, а также метод `__eq__()`, который в состоянии выполнять базовые сравнения объектов.

3. Как создать неизменяемый класс данных?

Чтобы сделать класс данных неизменяемым, установите `frozen=True` при создании.

Вывод: приобрел навыки по работе с классами данных при написании программ с помощью языка программирования Python версии 3.x.