Федеральное государственное бюджетное образовательное учреждение высшего образования Московский государственный технологический университет "ФГБОУ ВО МГТУ «СТАНКИН"

Кафедра «Управление и информатика в технических системах

Отчёт по лабораторной работе №1 по дисциплине "Прикладное программирование"

Выполнил:
Студент группы ИДБ-22-10
Инкин Денис Васильевич
Преподаватель:
Левченко А.Н.

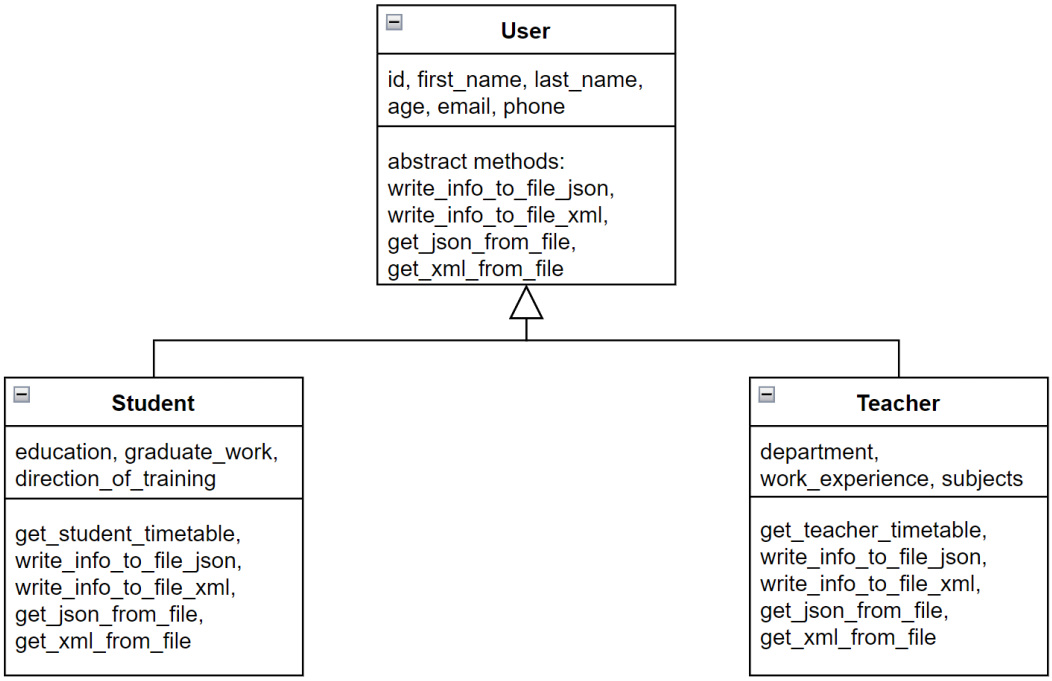
Москва, 2023

Содержание

1 Постановка задачи

- Реализовать базовый класс User с полями: id, first_name, last_name, age, email, phone
- Реализовать класс Student, который наследуется от класса User, со своими уникальными полями: education, graduate_work, direction_of_training
- Реализовать класс Teacher, которые наследуется от класса User, со своими уникальными полями: department, work_experience, subjects
- Для каждого дочернего класса реализовать несколько методов.
- Инофрмацию об объекте класса представить в формате json/xml. Добавить реализацию считывания из файла и запись в файл в соответствующих форматах (json/xml)
- Спроектировать в сервисе draw.io диаграмму классов
- Подготовить отчёт о выполненной лабораторной работе. Подготовиться по вопросам к защите.

2 Диаграмма классов



3 Код на python

Ссылка на GitHub: https://github.com/slain1gg/Application_programming.git

```
import json
from abc import ABC, abstractmethod
import xml.etree.ElementTree as ET


class AgeError(Exception):
    def __init__(self, age, message='должен быть от 0 до 150 лет'):
        self.age = age
        self.message = message
        super().__init__(self.message)

    def __str__(self):
        return f'Возраст {self.age} {self.message}'


class User(ABC):
    def __init__(self, id, first_name, last_name, age, email, phone):
        self._id = id
        self._first_name = first_name
        self._last_name = last_name
        self._email = email
        self._phone = phone
        if 0 <= age <= 150:
            self._age = age
        else:
            raise AgeError(age)

    @abstractmethod
    def write_info_to_file_json(self, file):
        pass

    @abstractmethod
    def get_json_from_file(self, file):
        pass

    @abstractmethod
    def write_info_to_file_xml(self, file):
```

```
        pass

    @abstractmethod
    def get_xml_from_file(self, file):
        pass


class Student(User):
    def __init__(self, id, first_name, last_name, age, email, phone,\
    education, graduate_work, direction_of_training):
        super().__init__(id, first_name, last_name, age, email, phone)
        self.__education = education
        self.__graduate_work = graduate_work
        self.__direction_of_training = direction_of_training

    def get_students_timetable(self):
        print('Получено студенческое расписание')

    def write_info_to_file_json(self, file):
        info = {
            'id': self._id,
            'first_name': self._first_name,
            'last_name': self._last_name,
            'age': self._age,
            'email': self._email,
            'phone': self._phone,
            'education': self.__education,
            'graduate_work': self.__graduate_work,
            'direction_of_training': self.__direction_of_training
        }
        with open(file, 'w') as f:
            json.dump(info, f, indent=4)

    def get_json_from_file(self, file):
        try:
            with open(file, 'r') as f:
                info = json.load(f)
            return info
        except FileNotFoundError:
            print('Ошибка при вызове функции get_json_from_file. \
```

```
            ТАКОГО ФАЙЛА НЕ НАЙДЕНО! ПРОВЕРЬТЕ ПУТЬ!!!!')

    @property
    def age(self):
        return self._age

    def get_xml_from_file(self, file):
        tree = ET.parse(file)
        root = tree.getroot()
        for elem in root:
            for subelem in elem:
                print(f"{subelem.tag}:{subelem.text}")

    def write_info_to_file_xml(self, file):
        info = {
            'id': self._id,
            'first_name': self._first_name,
            'last_name': self._last_name,
            'age': self._age,
            'email': self._email,
            'phone': self._phone,
            'education': self.__education,
            'graduate_work': self.__graduate_work,
            'direction_of_training': self.__direction_of_training
        }
        root = ET.Element('root')

        teacher = ET.SubElement(root, 'teacher')
        for key in info:
            ET.SubElement(teacher, key).text = str(info[key])
        tree = ET.ElementTree(root)
        tree.write(file)


class Teacher(User):
    def __init__(self, id, first_name, last_name, age, email, phone, \
    department, work_experience, subjects):
        super().__init__(id, first_name, last_name, age, email, phone)
        self.__department = department
        self.__work_experience = work_experience
```

```
        self.__subjects = subjects

    def get_teacher_timetable(self):
        print('Получено расписание для преподавателя')

    def write_info_to_file_json(self, file):
        info = {
            'id': self._id,
            'first_name': self._first_name,
            'last_name': self._last_name,
            'age': self._age,
            'email': self._email,
            'phone': self._phone,
            'department': self.__department,
            'work_experience': self.__work_experience,
            'subjects': self.__subjects
        }
        with open(file, 'w') as f:
            json.dump(info, f, indent=4)

    def get_json_from_file(self, file):
        try:
            with open(file, 'r') as f:
                info = json.load(f)
            return info
        except FileNotFoundError:
            print('Ошибка при вызове функции get_json_from_file. \
            ТАКОГО ФАЙЛА НЕ НАЙДЕНО! ПРОВЕРЬТЕ ПУТЬ!!!')

    @property
    def age(self):
        return self._age

    def get_xml_from_file(self, file):
        tree = ET.parse(file)
        root = tree.getroot()
        for elem in root:
            for subelem in elem:
                print(f"{subelem.tag}:{subelem.text}")
```

```
    def write_info_to_file_xml(self, file):
        info = {
            'id': self._id,
            'first_name': self._first_name,
            'last_name': self._last_name,
            'age': self._age,
            'email': self._email,
            'phone': self._phone,
            'department': self.__department,
            'work_experience': self.__work_experience,
            'subjects': self.__subjects
        }
        root = ET.Element('root')
        teacher = ET.SubElement(root, 'teacher')
        for key in info:
            ET.SubElement(teacher, key).text = str(info[key])
        subjects = ET.SubElement(teacher, 'subjects')
        for i in range(len(info['subjects'])):
            ET.SubElement(subjects, f'subject{i+1}').text = \
            str(info['subjects'][i])
        tree = ET.ElementTree(root)
        tree.write(file)


Elkin = Teacher(0, 'Alexandr', 'Elkin', 30, 'El@gmail.com', '+79084445569', 'Math
Test = Student(0, 'Alexandr', 'Elkin', 30, 'El@gmail.com', '+79084445569', 1,1,1)
Test.get_xml_from_file('files/test2.xml')
```