МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ   
ФЕДЕРАЦИИ

ФГБОУ ВО «БАЙКАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра математических методов и цифровых технологий

ОТЧЕТ

|  |  |  |
| --- | --- | --- |
| Студента бакалавриата группы | БИ-19-1 | Барлуков Н.В |

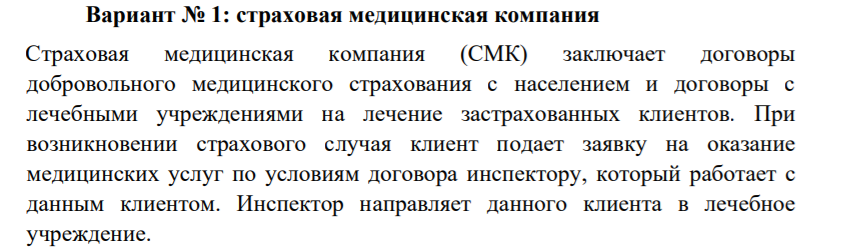
|  |  |  |
| --- | --- | --- |
| Руководитель(-и)  от университета |  | Родионов А.В |

Иркутск, 2021

**Лабораторная работа №3**

**Объектно-ориентированное программирование**

**1 Вариант**



Код

Класс Client

class Client:

    def \_\_init\_\_(self, fio, id, passport, phonenumber, email, dateofbirth, occupation):

        self.fio = fio

        self.passport = passport

        self.phonenumber = phonenumber

        self.email = email

        self.id = id

        self.dateofbirth = dateofbirth

        self.occupation = occupation

    def get\_string(self):

        return f'''

        ИД: {self.id}

        ФИО: {self.fio}

        Телефон: {self.phonenumber}

        Email: {self.email}

        Паспорт: {self.passport}

        Дата рождения: {self.dateofbirth}

        Род деятельности: {self.occupation}

        '''

Класс Clients

import json

from Models.Client import Client

import pandas as pandas

import os, json

from Utilities import Utilities as utils

class Clients:

    \_\_dir = os.path.dirname(\_\_file\_\_)

    \_\_filename = os.path.join(\_\_dir,'../', 'Source/clients.json')

    def \_\_init\_\_(self):

        self.\_\_clients = list()

    def \_\_len\_\_(self):

        return len(self.\_\_clients)

    def \_\_iter\_\_(self):

        return ClientsIterator(self.\_\_clients)

    def add(self, client : Client):

        self.\_\_clients.append(client)

        return client.id

    def get\_by\_id(self, id : int):

        for client in self.\_\_clients:

            if client.id == id:

                return client

        return None

    def delete(self, id : int):

        client = self.get\_by\_id(id)

        if (client):

            self.\_\_clients.remove(client)

            return True

        else:

            return False

    def edit(self, id : int, client : Client):

        cl = self.get\_by\_id(id)

        if(cl):

            cl.fio = client.fio

            cl.phonenumber = client.phonenumber

            cl.occupation = client.occupation

            cl.dateofbirth = client.dateofbirth

            cl.email = client.email

            cl.passport = client.passport

            return True

        else:

            return False

    def save(self):

        utils.save(self.\_\_filename, self.\_\_clients)

    def read(self):

        self.\_\_clients = list()

        file = open(self.\_\_filename, 'r', encoding='utf-8')

        data = json.loads(file.read())

        for elem in data:

            client = Client(elem['fio'], elem['id'],  elem['passport'], elem['phonenumber'],

                elem['email'], elem['dateofbirth'], elem['occupation'])

            self.add(client)

    def get\_new\_id(self):

        max = 0

        for el in self.\_\_clients:

            if el.id > max:

                max = el.id

        return max + 1

    def get\_clients\_table(self):

        id\_arr = []

        fio\_arr = []

        phone\_arr = []

        email\_arr = []

        passport\_arr = []

        date\_of\_birth\_arr = []

        occupation\_arr = []

        for elem in self.\_\_clients:

            id\_arr.append(elem.id)

            fio\_arr.append(elem.fio)

            phone\_arr.append(elem.phonenumber)

            email\_arr.append(elem.email)

            passport\_arr.append(elem.passport)

            date\_of\_birth\_arr.append(elem.dateofbirth)

            occupation\_arr.append(elem.occupation)

        return pandas.DataFrame(

            {"ИД": id\_arr,

            "ФИО": fio\_arr,

            "Номер телефона": phone\_arr,

            "Email": email\_arr,

            "Паспорт": passport\_arr,

            "Дата рождения": date\_of\_birth\_arr,

            "Род деятельности": occupation\_arr},

            index=None)

class ClientsIterator:

    def \_\_init\_\_(self, clients):

        self.\_\_clients = clients

        self.\_index = 0

    def \_\_next\_\_(self):

        try:

            result = self.\_\_clients[self.\_index]

            self.\_index +=1

            return result

        except IndexError:

           self.\_index = 0

           raise StopIteration

Класс Contraсt

from Models.Clients import Clients

from Models.Employees import Employees

from Models.Hospitals import Hospitals

from Models.Client import Client

class Contract:

    def \_\_init\_\_(self, id, clientId, employeeId, hospitalId, price, conclusionDate, status = False):

        self.id = id

        self.clientId = clientId

        self.employeeId = employeeId

        self.hospitalId = hospitalId

        self.price = price

        self.conclusionDate = conclusionDate

        self.status = status

    def get\_client(self, clients : Clients):

        return clients.get\_by\_id(self.clientId)

    def get\_employee(self, employees : Employees):

        return employees.get\_by\_id(self.employeeId)

    def set\_client(self, clientId):

        self.clientId = clientId

    def set\_employee(self,employeeId):

        self.employeeId = employeeId

    def get\_client\_str(self, clients: Clients):

        try:

            return self.get\_client(clients).fio

        except AttributeError:

            return ''

    def get\_employee\_str(self, employees : Employees):

        try:

            return self.get\_employee(employees).fio

        except AttributeError:

            return ''

    def get\_hospital(self, hospitals : Hospitals):

        return hospitals.get\_by\_id(self.hospitalId)

    def set\_hospital(self, hospitalId):

        self.hospitalId = hospitalId

    def get\_hospital\_str(self, hospitals : Hospitals):

        try:

            return self.get\_hospital(hospitals).name

        except AttributeError:

            return ''

    def get\_status(self):

        if not self.status:

            return "Договор оформлен"

        else:

            return "Договор исполнен"

    def set\_done(self):

        self.status = True

    def get\_string(self):

        status = self.get\_status()

        client = self.get\_client\_str(self.clientId)

        employee = self.get\_employee\_str(self.employeeId)

        hospital = self.get\_hospital\_str(self.hospitalId)

        return f'''

        Айди : {self.id}

        Клиент : {client}

        Сотрудник : {employee}

        Лечебное учреждение : {hospital}

        Цена : {self.price}

        Дата заключения договора : {self.conclusionDate}

        Статус : {status}

        '''

Класс Contracts

import json

import os, json

from Models.Contract import Contract

from Utilities import Utilities as utils

import pandas as pandas

class Contracts:

    \_\_dir = os.path.dirname(\_\_file\_\_)

    \_\_filename = os.path.join(\_\_dir,'../', 'Source/contracts.json')

    def \_\_init\_\_(self):

        self.\_\_contracts = list()

    def \_\_len\_\_(self):

        return len(self.\_\_contracts)

    def \_\_iter\_\_(self):

        return ContractsIterator(self.\_\_contracts)

    def add(self, contract: Contract):

        self.\_\_contracts.append(contract)

        return contract.id

    def delete(self, id: int):

        con = self.get\_by\_id(id)

        if con:

            self.\_\_contracts.remove(con)

            return True

        return False

    def edit(self, id: int, contract: Contract):

        con = self.get\_by\_id(id)

        if con:

            con.clientId = contract.clientId

            con.employeeId = contract.employeeId

            con.hospitalId = contract.hospitalId

            con.price = contract.price

            con.conclusionDate = contract.conclusionDate

            con.status = contract.status

            con.cost = contract.cost

            return True

        return False

    def get\_by\_id(self, id: int):

        for con in self.\_\_contracts:

            if con.id == id:

                return con

        return None

    def save(self):

        utils.save(self.\_\_filename, self.\_\_contracts)

    def read(self):

        self.\_\_contracts= list()

        file = open(self.\_\_filename, 'r', encoding='utf-8')

        data = json.loads(file.read())

        for elem in data:

            contract = Contract(elem['id'], elem['clientId'], elem['employeeId'], elem['hospitalId'],

                elem['price'], elem['conclusionDate'], elem['status'])

            self.add(contract)

    def get\_contracts\_table(self, hospitals, clients, employees):

        id\_arr = []

        client\_arr = []

        employee\_arr = []

        hospital\_arr = []

        price\_arr = []

        conclusionDate\_arr = []

        status\_arr = []

        for elem in self.\_\_contracts:

            id\_arr.append(elem.id)

            client\_arr.append(elem.get\_client\_str(clients))

            employee\_arr.append(elem.get\_employee\_str(employees))

            hospital\_arr.append(elem.get\_hospital\_str(hospitals))

            price\_arr.append(elem.price)

            conclusionDate\_arr.append(elem.conclusionDate)

            status\_arr.append(elem.get\_status())

        return pandas.DataFrame(

            {"Айди": id\_arr,

            "Клиент": client\_arr,

            "Сотрудник": employee\_arr,

            "Лечебное учреждение": hospital\_arr,

            "Цена": price\_arr,

            "Дата заключения договора": conclusionDate\_arr,

            "Статус": status\_arr,},

            index=None)

    def get\_done\_contracts\_table(self, hospitals, clients, employees):

        id\_arr = []

        client\_arr = []

        employee\_arr = []

        hospital\_arr = []

        price\_arr = []

        conclusionDate\_arr = []

        status\_arr = []

        for elem in self.\_\_contracts:

            if elem.status:

                id\_arr.append(elem.id)

                client\_arr.append(elem.get\_client\_str(clients))

                employee\_arr.append(elem.get\_employee\_str(employees))

                hospital\_arr.append(elem.get\_hospital\_str(hospitals))

                price\_arr.append(elem.price)

                conclusionDate\_arr.append(elem.conclusionDate)

                status\_arr.append(elem.get\_status())

        return pandas.DataFrame(

            {"Айди": id\_arr,

            "Клиент": client\_arr,

            "Сотрудник": employee\_arr,

            "Лечебное учреждение": hospital\_arr,

            "Цена": price\_arr,

            "Дата заключения договора": conclusionDate\_arr,

            "Статус": status\_arr,},

            index=None)

    def get\_new\_id(self):

        max = 0

        for el in self.\_\_contracts:

            if el.id > max:

                max = el.id

        return max + 1

class ContractsIterator:

   def \_\_init\_\_(self, contracts):

       self.\_\_contracts = contracts

       self.\_index = 0

   def \_\_next\_\_(self):

        try:

            result = self.\_\_contracts[self.\_index]

            self.\_index +=1

            return result

        except IndexError:

           self.\_index = 0

           raise StopIteration

Класс Employee

class Employee:

    def \_\_init\_\_(self, fio, id, passport, phonenumber, email, login, password):

        self.fio=fio

        self.id=id

        self.passport=passport

        self.phonenumber=phonenumber

        self.email=email

        self.login=login

        self.password=password

    def get\_string(self):

        return f'''

        ИД: {self.id}

        ФИО: {self.fio}

        Телефон: {self.phonenumber}

        Email: {self.email}

        Паспорт: {self.passport}

        Логин: {self.login}

        Пароль: {self.password}

        '''

    def set\_password(self, password):

        newPassword = input("Укажите новый пароль")

        self.password = newPassword

Класс Employees

import pandas as pandas

from Models.Employee import Employee

import json

import os, json

from Utilities import Utilities as utils

class Employees():

    \_\_dir = os.path.dirname(\_\_file\_\_)

    \_\_filename = os.path.join(\_\_dir,"../","Source/employees.json")

    def \_\_init\_\_(self):

        self.\_\_employees = list()

    def \_\_len\_\_(self):

        return len(self.\_\_employees)

    def \_\_iter\_\_(self):

        return EmployeesIterator(self.\_\_clients)

    def add(self, employee : Employee):

        self.\_\_employees.append(employee)

        return employee.id

    def get\_by\_id(self, id : int):

        for employee in self.\_\_employees:

            if(employee.id == id):

                return employee

        return None

    def delete(self, id : int):

        employee = self.get\_by\_id(id)

        if(employee):

            self.\_\_employees.remove(employee)

            return True

        else:

            return False

    def edit(self, id : int, employee : Employee):

        emp = self.get\_by\_id(id)

        if(emp):

            emp.fio = employee.fio

            emp.phonenumber = employee.phonenumber

            emp.login= employee.login

            emp.email = employee.email

            emp.password = employee.passport

            emp.passport = employee.passport

            return True

        else:

            return False

    def save(self):

        utils.save(self.\_\_filename, self.\_\_employees)

    def read(self):

        self.\_\_employees = list()

        file = open(self.\_\_filename, 'r', encoding='utf-8')

        data = json.loads(file.read())

        for elem in data:

            employee = Employee(elem['fio'], elem['id'], elem['passport'], elem['phonenumber'], elem['email'],

                elem['login'], elem['password'])

            self.add(employee)

    def get\_new\_id(self):

        max = 0

        for el in self.\_\_employees:

            if (el.id > max):

                max = el.id

        return max + 1

    def get\_employess\_table(self):

        id\_arr = []

        fio\_arr = []

        phone\_arr = []

        email\_arr = []

        password\_arr = []

        login\_arr = []

        passport\_arr = []

        for elem in self.\_\_employees:

            id\_arr.append(elem.id)

            fio\_arr.append(elem.fio)

            phone\_arr.append(elem.phonenumber)

            email\_arr.append(elem.email)

            password\_arr.append(elem.password)

            login\_arr.append(elem.login)

            passport\_arr.append(elem.passport)

        return pandas.DataFrame(

            {"ИД": id\_arr,

            "ФИО": fio\_arr,

            "Номер телефона": phone\_arr,

            "Email": email\_arr,

            "Пароль": password\_arr,

            "Логин": login\_arr,

            "Паспортные данные": passport\_arr},

            index=None)

class EmployeesIterator:

    def \_\_init\_\_(self, employees):

        self.\_\_employees = employees

        self.\_index = 0

    def \_\_len\_\_(self):

        return self.\_\_employees

    def \_\_next\_\_(self):

        try:

            result = self.\_\_employees[self.\_index]

            self.\_index +=1

            return result

        except IndexError:

           self.\_index = 0

           raise StopIteration

Класс Hospital

class Hospital:

    def \_\_init\_\_(self, id , address, name, phonenumber, email):

        self.id = id

        self.address = address

        self.name = name

        self.phonenumber = phonenumber

        self.email = email

    def get\_string(self):

        return f'''

        ИД: {self.id}

        Адрес: {self.adress}

        Телефон: {self.phonenumber}

        Email: {self.email}

        '''

Класс Hospitals

import pandas as pandas

from Models.Hospital import Hospital

import json

import os, json

from Utilities import Utilities as utils

class Hospitals:

    \_\_dir = os.path.dirname(\_\_file\_\_)

    \_\_filename = os.path.join(\_\_dir,"../","Source/hospitals.json")

    def \_\_init\_\_(self):

        self.\_\_hospitals = list()

    def \_\_len\_\_(self):

        return len(self.\_\_hospitals)

    def add(self, hospital : Hospital):

        self.\_\_hospitals.append(hospital)

        return hospital.id

    def get\_by\_id(self, id : int):

        for hosp in self.\_\_hospitals:

            if(hosp.id == id):

                return hosp

        return None

    def delete(self, id : int):

        hosp = self.get\_by\_id(id)

        if (hosp):

            self.\_\_hospitals.remove(hosp)

            return True

        else:

            return False

    def edit(self, id : int, hospital : Hospital):

        hosp = self.get\_by\_id(id)

        if(hosp):

            hosp.address = hospital.adress

            hosp.name = hospital.name

            hosp.phonenumber = hospital.phonenumber

            hosp.email = hospital.email

            return True

        else:

            return False

    def save(self):

        utils.save(self.\_\_filename,self.\_\_hospitals)

    def read(self):

        self.\_\_hospitals = list()

        file = open(self.\_\_filename, 'r', encoding='utf-8')

        data = json.loads(file.read())

        for elem in data:

            hospital = Hospital(elem['id'], elem['address'], elem['name'], elem['phonenumber'],

                 elem['email'])

            self.add(hospital)

    def get\_new\_id(self):

        max = 0

        for hosp in self.\_\_hospitals:

            if(hosp.id > max):

                max = hosp.id

        return max + 1

    def get\_hospitals\_table(self):

        id\_arr = []

        address\_arr = []

        name\_arr = []

        phonenumber\_arr = []

        email\_arr = []

        for elem in self.\_\_hospitals:

            id\_arr.append(elem.id)

            address\_arr.append(elem.address)

            name\_arr.append(elem.name)

            phonenumber\_arr.append(elem.phonenumber)

            email\_arr.append(elem.email)

        return pandas.DataFrame(

            {"ИД": id\_arr,

            "Адрес": address\_arr,

            "Название": name\_arr,

            "Номер телефона": phonenumber\_arr,

            "Электронная почта": email\_arr,},

            index=None)

class HospitalsIterator:

    def \_\_init\_\_(self, hospitals):

        self.\_\_hospitals = hospitals

        self.\_index = 0

    def \_\_next\_\_(self):

        try:

            result = self.\_\_hospitals[self.\_index]

            self.\_index +=1

            return result

        except IndexError:

           self.\_index = 0

           raise StopIteration

Класс Utilities

import json

def save(path, collection):

    class ComplexEncoder(json.JSONEncoder):

        def default(self, z):

            if isinstance(z, complex):

                return (z.real, z.imag)

            return super().default(z)

    with open(path, 'w', encoding='utf-8') as write:

        json.dump(collection, write, default=lambda o: o.\_\_dict\_\_,

            cls=ComplexEncoder)

Пример выполнения

