Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 5 з дисципліни

«Основи програмування – 2. Метидології програмування»

«Наслідування та поліморфізм» Варіант 29

Виконав студент ІП-13 Романюк Діана Олексіївна

(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна

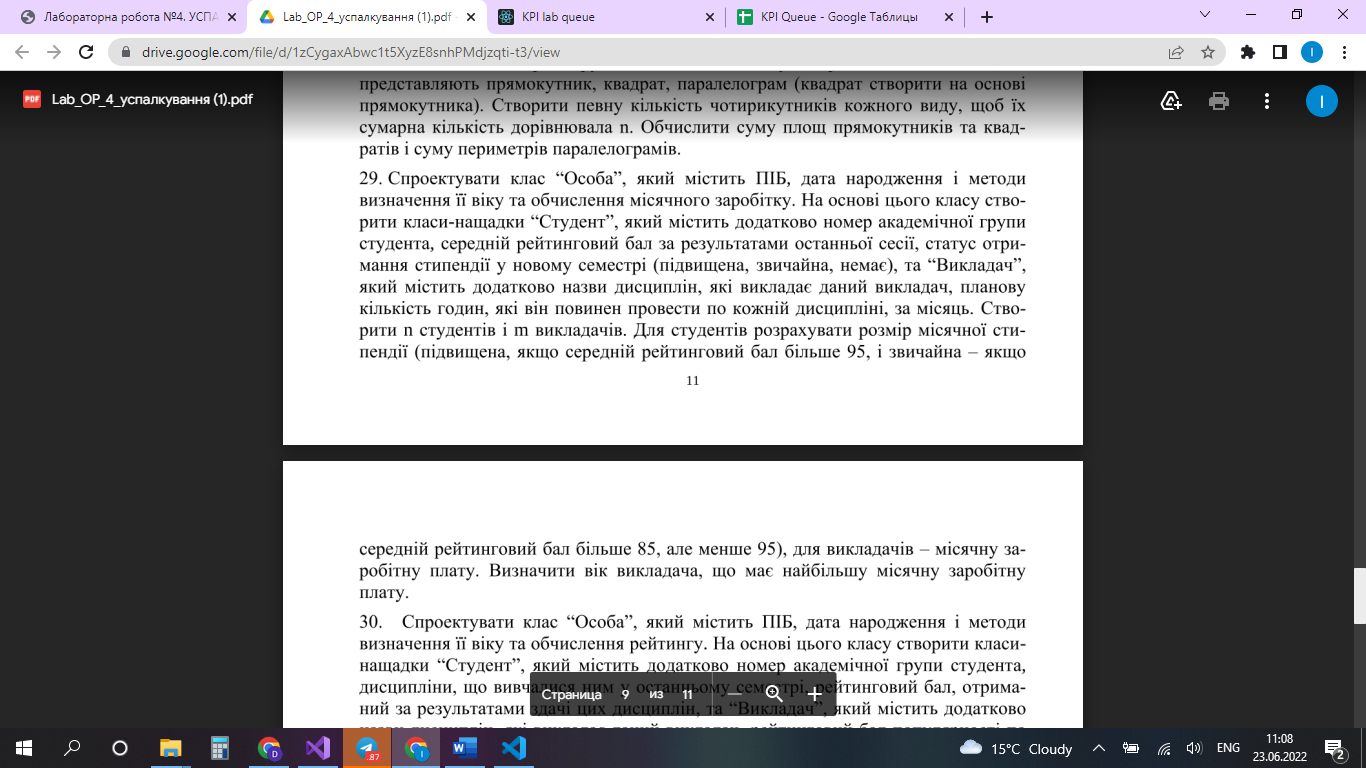
( прізвище, ім'я, по батькові)

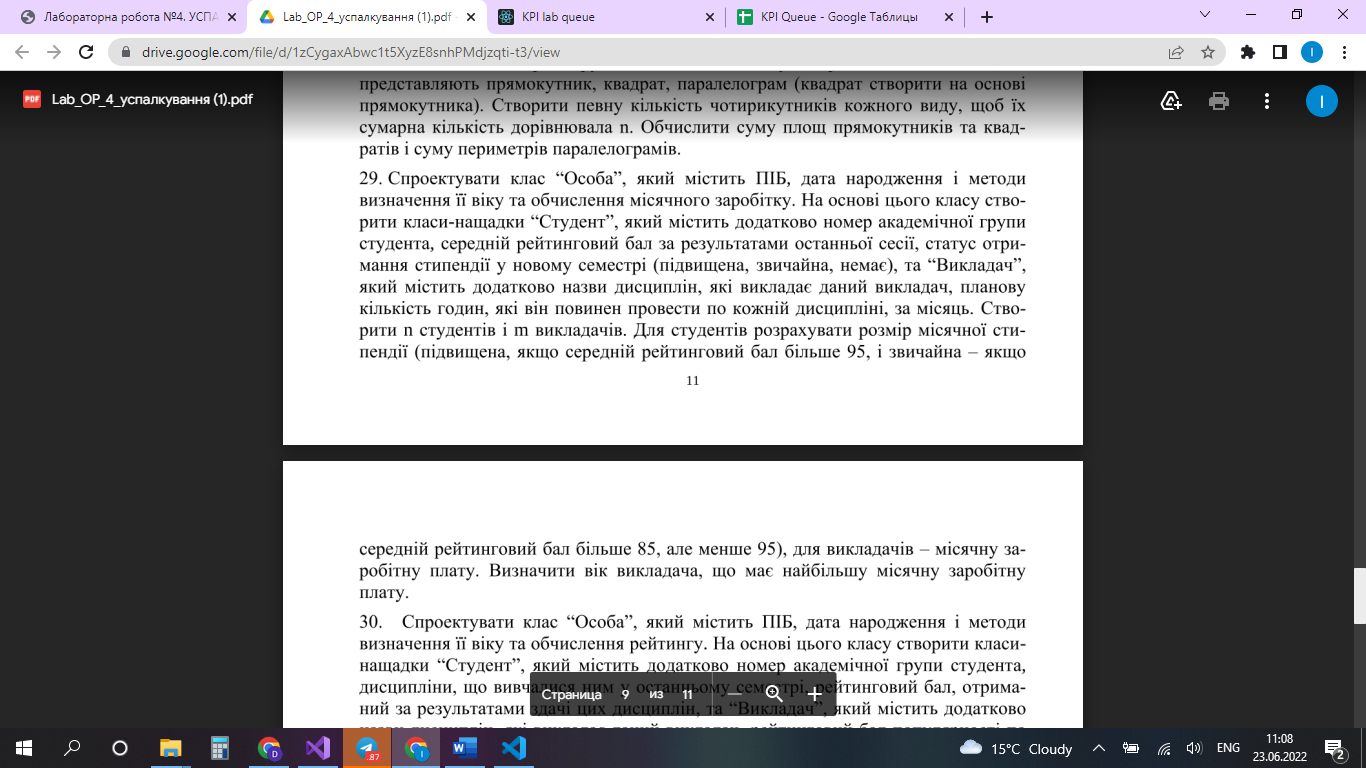
Київ 2022

# Лабораторна робота 5

**Мета** – вивчити механізми створення класів та об’єктів.

# Варіант 29 Постановка задачі:





**Код програми на С++:**

# Header.h

#include <iostream>

#include <string>

#include <vector>

#include <Windows.h>

using namespace std;

class Person {

private:

string name;

string date\_of\_birth;

public:

Person(string line);

string getName();

string count\_age();

virtual double monthly\_income(double);

};

class Student : public Person {

private:

int group\_id;

double average\_grade;

char status;

public:

Student(string line1 = "none none none 01:01:1970", string line2 = "00 0.0");

double monthly\_income(double) override;

};

class Teacher : public Person {

private:

string subject;

int hours;

public:

int getHours();

double monthly\_income(double) override;

Teacher(string line1 = "none none none 01:01:1970", string line2 = "none 0");

};

# std::vector<string> split(string, char sep = ' ');

# Lab5.cpp

#include "Header.h"

int main() {

vector <Student> base\_of\_students;

vector <Teacher> base\_of\_teachers;

int n, m;

string str1, str2;

cout << "Enter amount of students: "; cin >> n;

cin.ignore();

for (int i = 0; i < n; i++) {

cout << "[Surname Name Patronymic DD::MM:YYYY]\n";

getline(cin, str1);

cout << "[id grades]\n";

getline(cin, str2);

base\_of\_students.push\_back(Student(str1, str2));

}

cout << "Enter amount of teachers: "; cin >> m;

cin.ignore();

for (int i = 0; i < m; i++) {

cout << "[Surname Name Patronymic DD::MM:YYYY]\n";

getline(cin, str1);

cout << "[subject hours]\n";

getline(cin, str2);

base\_of\_teachers.push\_back(Teacher(str1, str2));

}

double grants; cout << "Enter the grants amount: ";

cin >> grants;

cout << "Students:" << endl;

for (auto& s : base\_of\_students) {

cout << s.getName() << ": " << s.monthly\_income(grants) << " UAH" << endl;

}

double salary; cout << "Enter the teachers' salary per hour: ";

cin >> salary;

cout << "Teachers:" << endl;

Teacher richest;

if (base\_of\_teachers.size() > 0) {

richest = base\_of\_teachers[0];

}

for (int i = 0; i < m; i++) {

cout << base\_of\_teachers[i].getName() << ": " << base\_of\_teachers[i].monthly\_income(salary) << " UAH" << endl;

if (base\_of\_teachers[i].monthly\_income(salary) > richest.monthly\_income(salary)) richest = base\_of\_teachers[i];

}

cout << "The teacher with the biggest salary : " << richest.getName() << "\nHis age: " << richest.count\_age();

return 0;

}

# Header.cpp

#include "Header.h"

Person::Person(string line) {

name = split(line)[0] + ' ' + split(line)[1] + ' ' + split(line)[2];

date\_of\_birth = split(line)[3];

}

string Person::getName() {

return name;

}

double Person::monthly\_income(double hours) {

return (150 \* hours);

}

string Person::count\_age() {

SYSTEMTIME current\_date;

GetLocalTime(&current\_date);

int birth\_year = stoi(split(date\_of\_birth, ':')[2]);

int birth\_month = stoi(split(date\_of\_birth, ':')[1]);

int birth\_day = stoi(split(date\_of\_birth, ':')[0]);

int d\_day, d\_month, d\_year;

d\_year = current\_date.wYear - birth\_year;

if (current\_date.wMonth < birth\_month) { d\_month = current\_date.wMonth + 12 - birth\_month; }

else { d\_month = current\_date.wMonth - birth\_month; }

if (current\_date.wDay < birth\_day) { d\_day = current\_date.wDay + 30 - birth\_day; }

else { d\_day = current\_date.wDay - birth\_day; }

if (current\_date.wMonth < birth\_month || current\_date.wMonth == birth\_month && current\_date.wDay < birth\_day) {

d\_year--;

}

string age = "";

age += to\_string(d\_year) + "year(s)" + to\_string(d\_month) + "month(s)" + to\_string(d\_day) + "day(s)\n";

return age;

}

Student::Student(string line1, string line2) : Person(line1) {

group\_id = stoi(split(line2)[0]);

average\_grade = stod(split(line2)[1]);

}

double Student::monthly\_income(double grants)

{

if (average\_grade >= 95) {

status = 'h';

return grants \* 1.4;

}

if (average\_grade >= 85) {

status = 'd';

return grants;

}

status = 'n';

return 0;

}

Teacher::Teacher(string line1, string line2) : Person(line1) {

subject = split(line2)[0];

hours = stoi(split(line2)[1]);

}

int Teacher::getHours() {

return hours;

}

double Teacher::monthly\_income(double payment)

{

return (payment \* this->getHours());

}

vector<string> split(string line, char sep) {

vector<string> words;

string temp\_word = "";

line += sep;

for (int i = 0; i < line.length(); i++) {

if (line[i] == sep) {

if (temp\_word.length() > 0) {

words.push\_back(temp\_word);

}

temp\_word = "";

}

else {

temp\_word += line[i];

}

}

return words;

# }

# Код програми на Python

# Lab5.py

from foo import \*

base\_of\_students = list()

base\_of\_teachers = list()

n = int(input('Enter amount of students: '))

for i in range(n):

    line1 = input('[Surname Name Patronymic DD::MM:YYYY]\n')

    line2 = input('[id grades]\n')

    base\_of\_students.append(Student(line1, line2))

m = int(input('Enter amount of teachers: '))

for i in range(m):

    line1 = input('[Surname Name Patronymic DD::MM:YYYY]\n')

    line2 = input('[subject hours]\n')

    base\_of\_teachers.append(Teacher(line1, line2))

grants = float(input('Enter the grants amount: '))

print('STUDENTS:')

for i in range(n):

    print(f'student {base\_of\_students[i].get\_name()} {base\_of\_students[i].monthly\_income(grants)} UAH')

if m > 0:

    payment = float(input('Enter the payment per hour: '))

    richest = base\_of\_teachers[0]

for i in range(m):

    print(f'teacher {base\_of\_teachers[i].get\_name()} {base\_of\_teachers[i].monthly\_income(payment)} UAH')

    if base\_of\_teachers[i].monthly\_income(payment) > richest.monthly\_income(payment):

        richest = base\_of\_teachers[i]

print(f'Richest teacher: {richest.get\_name()}')

print(f'His age: {richest.get\_age()}')

# foo.py

from datetime import date

from abc import ABC, abstractmethod

class Person(ABC):

    def \_\_init\_\_(self, line: str):

        self.\_\_name\_\_ = line.split()[0] + ' ' + line.split()[1] + ' ' + line.split()[2]

        self.\_\_date\_of\_birth = line.split()[3]

    def get\_name(self):

        return self.\_\_name\_\_

    def get\_age(self) -> str:

        today = date.today()

        b\_day = int(self.\_\_date\_of\_birth.split(':')[0])

        b\_month = int(self.\_\_date\_of\_birth.split(':')[1])

        b\_year = int(self.\_\_date\_of\_birth.split(':')[2])

        d\_year = today.year - b\_year

        if today.month < b\_month:

            d\_month = today.month + 12 - b\_month

        else:

            d\_month = today.month - b\_month

        if today.day < b\_day:

            d\_day = today.day + 30 - b\_day

        else:

            d\_day = today.day - b\_day

        if today.month < b\_month or today.month == b\_month and today.day < b\_day:

            d\_year -= 1

        age = f'{str(d\_year)} year(s) {str(d\_month)} month(s) {str(d\_day)} day(s)'

        return age

    @abstractmethod

    def monthly\_income(self, hours: float) -> float:

        return 150 \* hours

class Student(Person):

    def \_\_init\_\_(self, line1, line2):

        super().\_\_init\_\_(line1)

        self.group\_id = int(line2.split()[0])

        self.\_\_average\_grade = float(line2.split()[1])

        self.\_\_status = 'n'

    def monthly\_income(self, grants: float) -> float:

        if self.\_\_average\_grade >= 95:

            self.\_\_status = 'h'

            return grants\*1.4

        elif self.\_\_average\_grade >= 85:

            self.\_\_status = 'd'

            return grants

        else:

            return 0

class Teacher(Person):

    def \_\_init\_\_(self, line1, line2):

        super().\_\_init\_\_(line1)

        self.\_\_subject = line2.split()[0]

        self.hours = float(line2.split()[1])

    def monthly\_income(self, salary: float) -> float:

        return salary\*self.hours

# 

**Висновки:** під час лабораторної роботи ми вивчили механізми створення класів та об’єктів.