**мІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**нАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «лЬВІВСЬКА ПОЛІТЕХНІКА»**

**ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра ІСМ**

|  |
| --- |
| **2022** |



**ЗВІТ**

**Про виконання лабораторної роботи №5.1(I)**

**«Класи з опрацюванням виняткових ситуацій»**

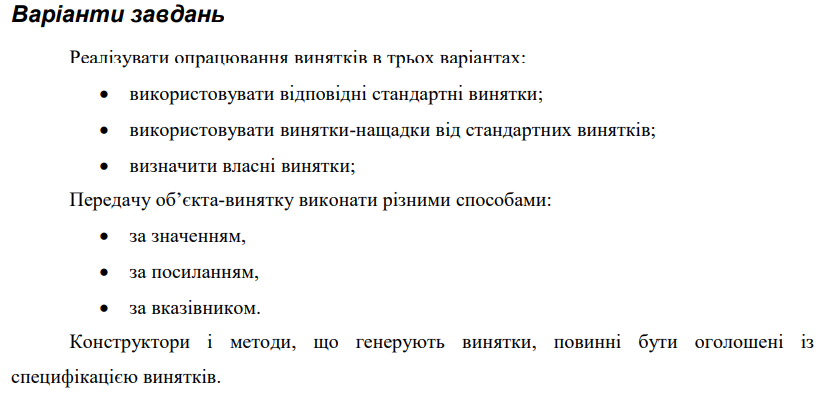
**З дисципліни**

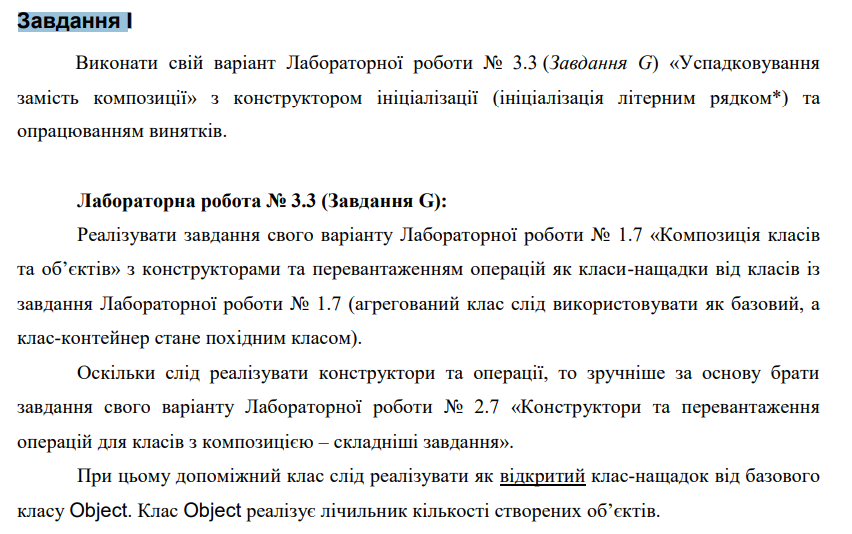
**«Об’єктно-орієнтоване програмування»**

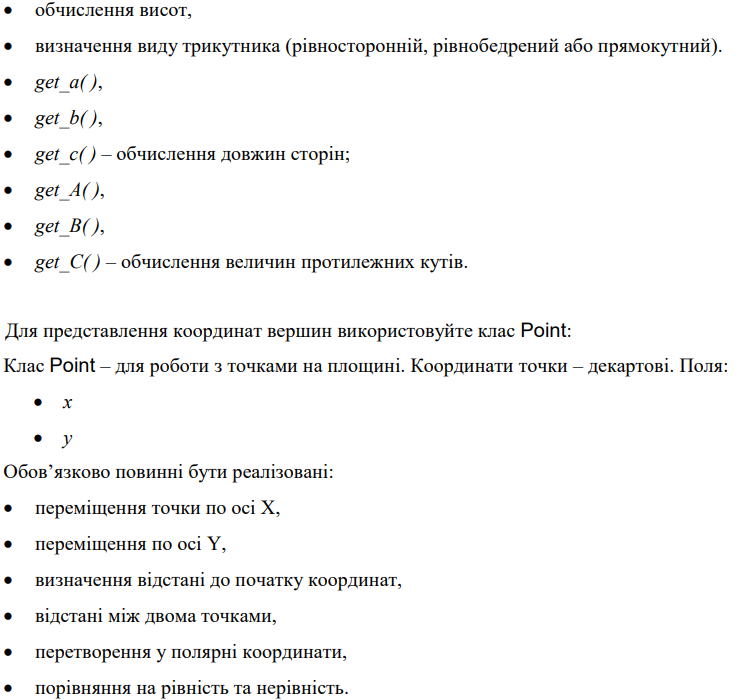
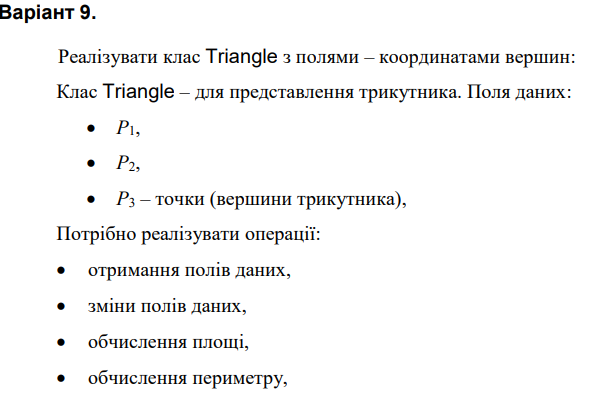
**Студента групи ІТ-11**

**Лисецького Володимира Любомировича**

**Умова завдання:**

****

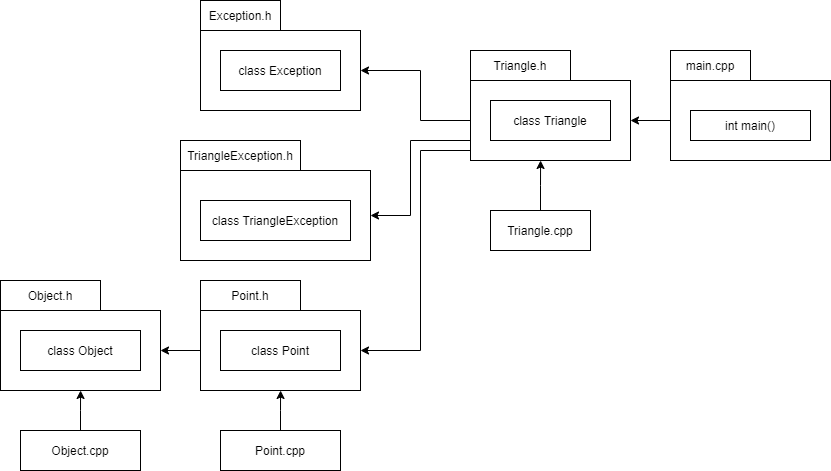
****

****

**UML-діаграма класів:**



**Структурна схема програми:**



**Текст програми:**

#pragma once

class Object

{

protected:

static unsigned int count;

public:

Object();

~Object();

unsigned int get\_count() { return count; };

};

///////////////////////////////////////////////////////////////

// Point.h

#pragma once

#include "Object.h"

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

class Point : public Object

{

private:

double x, y;

public:

double getX() { return x; }

double getY() { return y; }

void setX(double value) { x = value; }

void setY(double value) { y = value; }

Point();

Point(double a, double b);

Point(char\* a, char\* b);

Point(const Point& obj);

Point& operator = (const Point& obj);

operator string() const;

double distanceToOrigin();

void moveX(double value) { x += value; }

void moveY(double value) { y += value; }

void polarizeCoords();

friend istream& operator >> (istream& in, Point& obj);

friend ostream& operator << (ostream& in, const Point& obj);

Point& operator ++();

Point& operator --();

Point operator ++(int);

Point operator --(int);

friend double distanceOfPoints(Point p1, Point p2);

friend bool operator ==(Point p1, Point p2);

friend bool operator != (Point p1, Point p2);

Point getPoint() { return \*this; }

void setPoint(Point& obj) { \*this = obj; }

};

#include "Point.h"

#include "TriangleException.h"

#include "Exception.h"

#include <iostream>

#include <string>

class Triangle :

public Point

{

private:

Point P2, P3; // A B C

public:

Point getP2() { return P2; }

Point getP3() { return P3; }

void setP1(double x, double y) { setX(x); setY(y); }

void setP2(double x, double y) { P2.setX(x); P2.setY(y); }

void setP3(double x, double y) { P3.setX(x); P3.setY(y); }

Triangle();

Triangle(double x1, double y1, double x2, double y2, double x3, double y3);

Triangle(char\* x1, char\* y1, char\* x2, char\* y2, char\* x3, char\* y3);

operator string() const;

double getObjX(Point obj) const { return obj.getX(); }

double getObjY(Point obj) const { return obj.getY(); }

void setObjX(Point obj, double value) { obj.setX(value); }

void setObjY(Point obj, double value) { obj.setY(value); }

double get\_a()const { return distanceOfPoints(\*this, P2); }; // Side AB P1-P2 a

double get\_b() const { return distanceOfPoints(P2, P3); }; // Side BC P2-P3 b

double get\_c() const { return distanceOfPoints(\*this, P3); }; // Side AC P1-P3 c

double get\_A() const; // Angle A

double get\_B()const; // Angle B

double get\_C()const; // Angle C

double hA() const; // Height AH

double hB() const; // Height BH

double hC()const; // Height CH

friend istream& operator >> (istream& in, Triangle& obj);

friend ostream& operator << (ostream& in, const Triangle& obj);

double perimeter()const;

string triangleType()const;

double square() const;

};

#pragma once

#include<iostream>

#include<string>

#include<exception>

using namespace std;

class Exception : public exception

{

string message;

public:

Exception(string message) : message(message) {}

string What() { return message; }

};

#pragma once

#include<iostream>

#include<string>

using namespace std;

class TriangleException

{

string message;

public:

TriangleException(string message) : message(message) {}

string What() { return message; }

};

#include <iostream>

#include "Triangle.h"

int main()

{

try {

Triangle T;

cin >> T;

}

catch (TriangleException\* q) {

cout << "TriangleException: " << q->What() << endl;

}

catch (TriangleException& q) {

cout << "TriangleException: " << q.What() << endl;

}

catch (Exception q) {

cout << "Exception: " << q.What() << endl;

}

catch (invalid\_argument q) {

cout << "invalid\_argument: " << q.what() << endl;

}

}

**Посилання на git-репозиторій з проектом:**

**https://github.com/MarshmallowSoup/oop-lab-5-1**

**Результати unit-тесту:** 