МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 5

по дисциплине: ООП
тема: « Классы, виды отношений. Наследование.»

Выполнил: ст. группы ПВ202
Аладиб язан
Проверил:
Буханов Д.Г.

Белгород 2022

Цель работы: Получение теоретических знаний в области разработки классов, получение практических навыков реализаций классов и отношений между ними.

Задание:

В соответствии с вариантом (((номер по списку + 5)%10)+1) выполнить построение объектной модели (использовать не менее 5 объектов) заданной предметной области (задание 1), разработать диаграмму классов для описанной объектной модели (не менее 7 классов), и реализовать предложенные классы (задание 2).

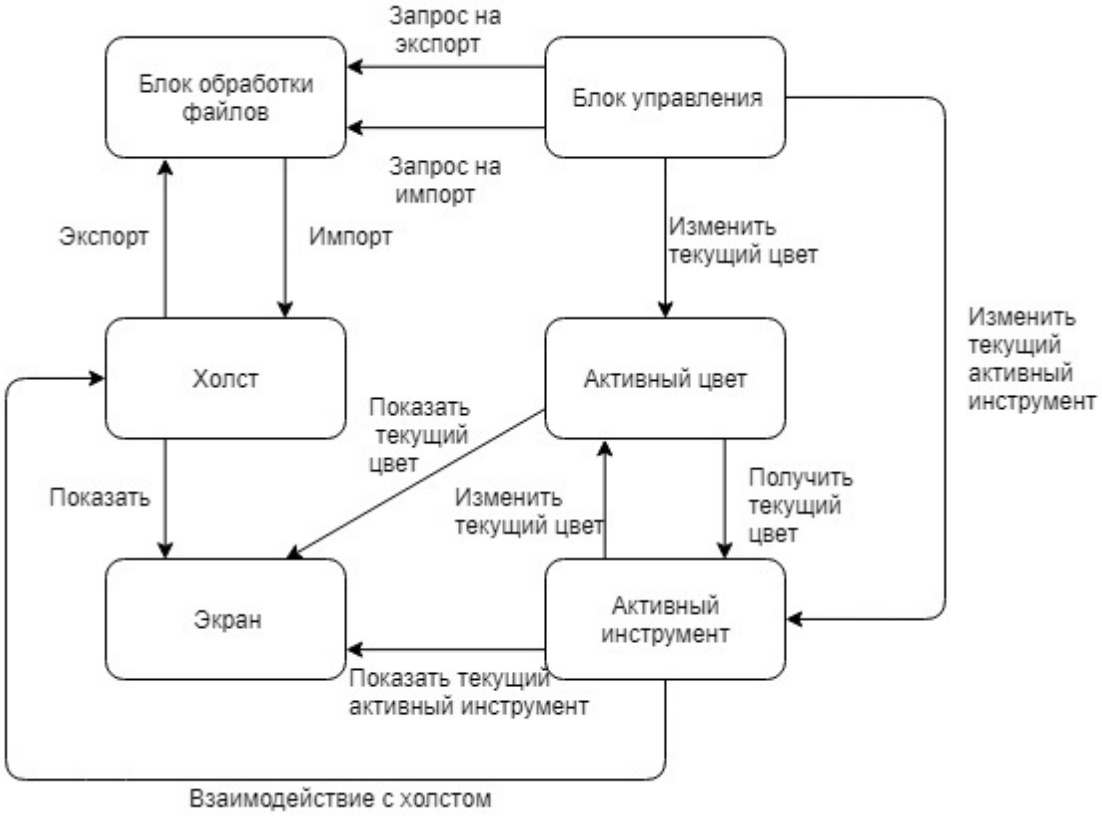
Задание Варианта :

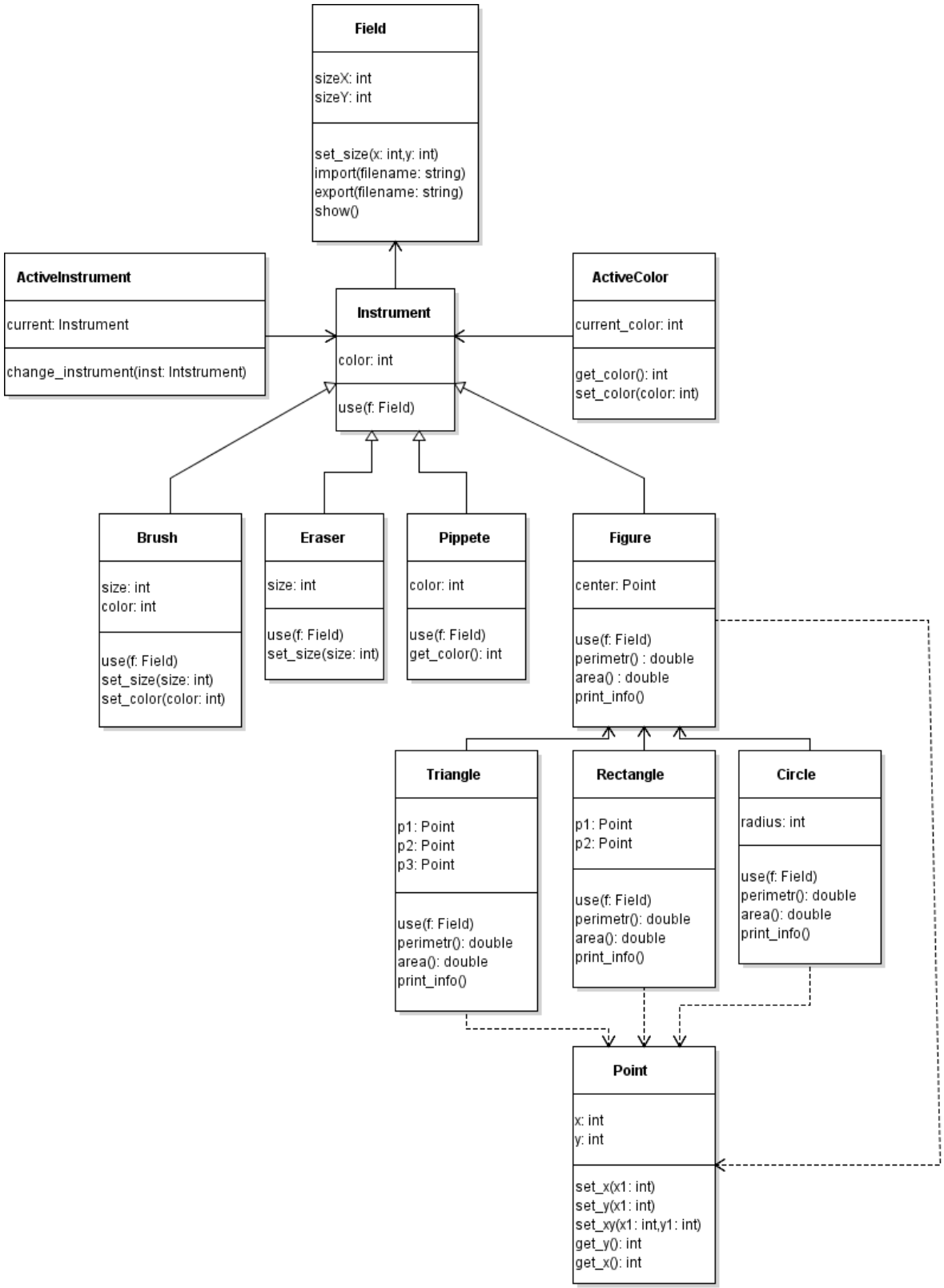
Задание 1
Графический редактор.

Задание 2
1. Создать абстрактный класс Figure с методами вычисления площади и периметра, а также методом, выводящим информацию о фигуре на экран.
2. Создать производные классы: Rectangle (прямоугольник), Circle (круг), Triangle (треугольник) со своими методами вычисления площади и периметра.
3. Создать массив n фигур и вывести полную информацию о фигурах на экран.

Выполнение работы:

Задание 1:




Field
sizeX: int
sizeY: int
set_size(x: int,y: int)
import(filename: string)
export(filename: string)
show()
ActiveInstrument
current: Instrument
change_instrument(inst: Intstrument)
Instrument
color: int
use(f: Field)
ActiveColor
current_color: int
get_color(): int
set_color(color: int)
Brush
size: int
color: int
use(f: Field)
set_size(size: int)
set_color(color: int)
Eraser
size: int
use(f: Field)
set_size(size: int)
Pippete
color: int
use(f: Field)
get_color(): int
Figure
center: Point
use(f: Field)
perimetr() : double
area() : double
print_info()
Triangle
p1: Point
p2: Point
p3: Point
use(f: Field)
perimetr(): double
area(): double
print_info()
Rectangle
p1: Point
p2: Point
use(f: Field)
perimetr(): double
area(): double
print_info()
Circle
radius: int
use(f: Field)
perimetr(): double
area(): double
print_info()
Point
x: int
y: int
set_x(x1: int)
set_y(x1: int)
set_xy(x1: int,y1: int)
get_y(): int
get_x(): int

Задание 2 :

```
#define _USE_MATH_DEFINES

#include <iostream>
#include <cmath>
#include <windows.h>

using namespace std;

class Point {
    double x, y;
public:
    Point(double x1, double y1) {
        this->x = x1;
        this->y = y1;
    }
    void set_x(double x1) { this->x = x1; };
    void set_y(double y1) { this->y = y1; };
    void set_xy(double x1, double y1) {
        this->set_x(x1);
        this->set_y(y1);
    };
    double get_x() { return this->x; };
    double get_y() { return this->y; };
};

class Figure {
    Point center;
public:
    Figure(Point t) : center(t) {};
    virtual double perimetr() = 0;
    virtual double area() = 0;
    virtual void print_info() = 0;
    virtual ~Figure() {};
};

class RRectangle : public Figure {
    Point p1, p2; // Diagonal points
public:
    RRectangle(Point a, Point b, Point center) : Figure(center), p1(a), p2(b) {};
    double perimetr();
    double area();
    void print_info();
};

class Triangle : public Figure {
    Point p1, p2, p3; // Triangle vertices
public:
```

```
    Triangle(Point a, Point b, Point c, Point center) : Figure(center), p1(a), p2(b),
p3(c) {};
    double perimetr();
    double area();
    void print_info();
};

class Circle : public Figure {
    double radius;
public:
    Circle(double r, Point center) : Figure(center), radius(r) {};
    double perimetr();
    double area();
    void print_info();
};

double point_distance(Point a, Point b) {
    double x = b.get_x() - a.get_x();
    double y = b.get_y() - a.get_y();
    return sqrt(x * x + y * y);
}

double RRectangle::perimetr() {
    Point temp(this->p1.get_x(), this->p2.get_y());
    return 2 * (point_distance(this->p1, temp) + point_distance(temp, this->p2));
};
double RRectangle::area() {
    Point temp(this->p1.get_x(), this->p2.get_y());
    return point_distance(this->p1, temp) * point_distance(temp, this->p2);
};
void RRectangle::print_info() {
    std::cout << "\nRectangle Information: " << std::endl;
    std::cout << "Perimeter = " << this->perimetr() << std::endl;
    std::cout << "Area = " << this->area() << std::endl << std::endl;
};

double Triangle::perimetr() {
    return point_distance(p1, p2) + point_distance(p2, p3) + point_distance(p1, p3);
};
double Triangle::area() {
    double a = point_distance(p1, p2);
    double b = point_distance(p2, p3);
    double c = point_distance(p1, p3);
    double p = 0.5 * (a + b + c);
    return sqrt(p * (p - a) * (p - b) * (p - c));
};
void Triangle::print_info() {
    std::cout << "\nTriangle Information: " << std::endl;
    std::cout << "Perimeter = " << this->perimetr() << std::endl;
    std::cout << "Area = " << this->area() << std::endl << std::endl;
};
```

```
double Circle::perimetr() {
    return 2 * radius * M_PI;
};
double Circle::area() {
    return M_PI * radius * radius;
};
void Circle::print_info() {
    std::cout << "\nCircle Information: " << std::endl;
    std::cout << "Radius = " << this->radius << std::endl;
    std::cout << "Perimeter = " << this->perimetr() << std::endl;
    std::cout << "Area = " << this->area() << std::endl << std::endl;
};

int main() {
    SetConsoleOutputCP(CP_UTF8);// Enable Russian language output
    char temp_type;
    double x1, x2, x3, y1, y2, y3, y4, x4;
    int n;
    cout << "Enter the number of shapes: ";
    cin >> n;
    Figure* arr[n];
    int i = 0;
    while (i < n) {
        cout << "Select the shape type (1-rectangle, 2-circle, 3-triangle): ";
        cin >> temp_type;

        switch (temp_type) {
            case '1':
                cout << "Enter center coordinates: ";
                cin >> x1 >> y1;
                cout << "Enter p1 coordinates: ";
                cin >> x2 >> y2;
                cout << "Enter p2 coordinates: ";
                cin >> x3 >> y3;
                arr[i] = new RRectangle(Point(x2, y2), Point(x3, y3), Point(x1, y1));
                i++;
                break;
            case '2':
                cout << "Enter center coordinates: ";
                cin >> x1 >> y1;
                cout << "Enter radius: ";
                cin >> x2;
                arr[i] = new Circle(x2, Point(x1, y1));
                i++;
                break;
            case '3':
                cout << "Enter center coordinates: ";
                cin >> x1 >> y1;
                cout << "Enter p1 coordinates: ";
                cin >> x2 >> y2;
```

```
                cout << "Enter p2 coordinates: ";
                cin >> x3 >> y3;
                cout << "Enter p3 coordinates: ";
                cin >> x4 >> y4;
                arr[i] = new Triangle(Point(x2, y2), Point(x3, y3), Point(x4, y4),
Point(x1, y1));
                i++;
                break;
            default:
                cout << "\nInvalid shape type." << endl;
                break;
        }
    }
    for (i = 0; i < n; i++) {
        arr[i]->print_info();
        delete arr[i];
    }
    return 0;
}
```