МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе№2

по дисциплине «Объектно-ориентированное программирование»

Тема: «Наследование»

Студент гр. 7382 Глазунов С.А.

Преподаватель Жангиров Т. Р.

Санкт-Петербург

2019

Цель работы.

Ознакомиться с понятиями наследование, полиморфизм, абстрактный класс, изучить виртуальные функции, принцип их работы, способ организации в памяти, раннее и позднее связывания в языке C++. В соответствии с индивидуальным заданием разработать систему классов для представления геометрических фигур.

Задание.

Необходимо спроектировать систему классов для моделирования геометрических фигур (в соответствии с полученным индивидуальным заданием). Задание предполагает использование виртуальных функций в иерархии наследования, проектирование и использование абстрактного базового класса. Разработанные классы должны быть наследниками абстрактного класса Shape, содержащего методы для перемещения в указанные координаты, поворота на заданный угол, масштабирования на заданный коэффициент, установки и получения цвета, а также оператор вывода в поток.

Необходимо также обеспечить однозначную идентификацию каждого объекта.

Решение должно содержать:

- Условие задания;
- UML диаграмму разработанных классов;
- Текстовое обоснование проектных решений;
- Реализацию классов на языке C++.

Индивидуальное задание.

Вариант 4– реализовать систему классов для фигур:

1. Круг;

2. Пятиконечная звезда;

3. Шестиконечная звезда.

UML диаграмма разработанных классов.

UML диаграмма разработанных классов представлена в приложении А.

Реализация классов на языке C++.

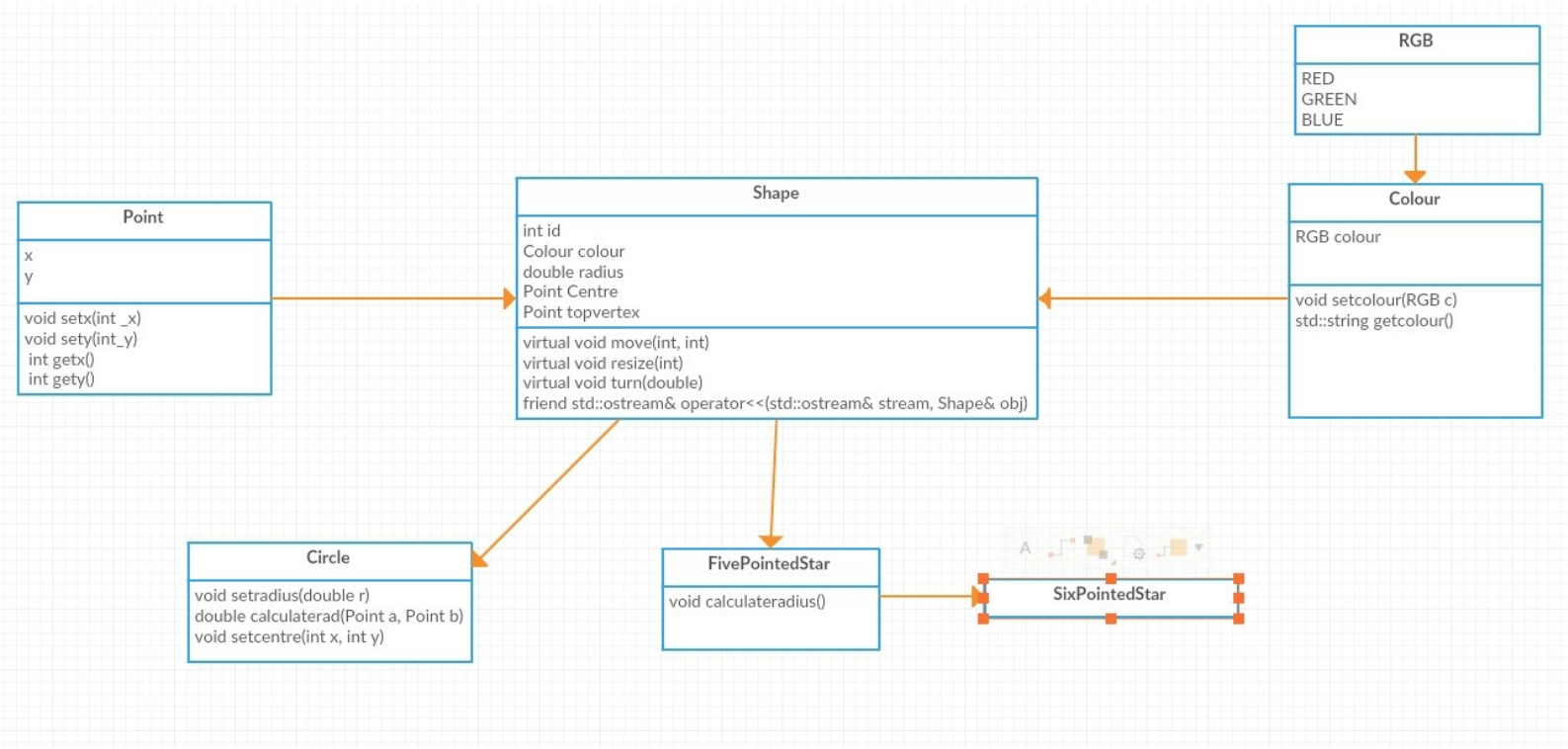
Реализация классов представлена в приложении Б.

Выводы.

В ходе выполнения лабораторной работы была спроектирована система классов для работы с геометрическими фигурами в соответствии с индивидуальным заданием. В иерархии наследования были использованы виртуальные функции, базовый класс при этом является виртуальным. Были реализованы методы перемещения фигуры в заданные координаты, поворота на заданный угол, масштабирования на заданный коэффициент, была реализована однозначная идентификация объекта.

ПРИЛОЖЕНИЕ А

UML ДИАГРАММА КЛАССОВ



ПРИЛОЖЕНИЕ Б

РЕАЛИЗАЦИЯ КЛАССОВ НА ЯЗЫКЕ C++

```
#include <iostream>
#include <cmath>
#include <string>
#define STANDART_CENTRE 0
#define STANDART_TOP 1
enum RGB { RED, GREEN, BLUE};
static int general_id;
class Colour
{
private:
RGB colour;
public:
void setcolour(RGB c)
{
switch (c)
{
case RED:
colour = RED;
break;
case GREEN:
colour = GREEN;
break;
case BLUE:
colour = BLUE;
break;
default:
break;
}
}

std::string getcolour()
{
switch (colour)
{
case RED:
return "Red";
case GREEN:
return "Green";
case BLUE:
return "Blue";
default:
return "";
}
}
```

```
};

//одна точка
class Point
{
private:
int x, y;

public:
Point()
{
x = 0;
y = 0;
}
Point(int _x, int _y)
{
x = _x;
y = _y;
}
void setx(int _x)
{
x = _x;
}
void sety(int _y)
{
y = _y;
}
int getx()
{
return x;
}
int gety()
{
return y;
}
};

//круг
//пятиконечная звезда
//шестиконечная звезда
//any figure

class Shape
{
public:
Shape()
{
id = general_id;
general_id++;
}
```

```
int id;
Point centre;
double radius;
Point topvertex;
Colour colour;
virtual void move(int, int) = 0;
virtual void resize(int) = 0;
virtual void turn(double) = 0;
friend std::ostream& operator<<(std::ostream& stream, Shape& obj);
};
std::ostream& operator<<(std::ostream& stream, Shape& object)
{
stream << "figure id:" << object.id << std::endl
<< "Color:" << object.colour.getcolour() << std::endl
<< "Centre: (" << object.centre.getx() << ", "<<object.centre.gety() << ")" << std::endl
<< "Another important point: (" << object.topvertex.getx() << ", " << object.topvertex.gety()
<< ")" << std::endl
<< "Radius :" << object.radius << std::endl;
return stream;
}

class Circle : public Shape
{
void setradius(double r)
{
radius = r;
}
//вычисляет радиус по центру и любой точке, лежащей на окружности
double calculaterad(Point a, Point b)
{
return sqrt((a.getx() - b.getx()) * (a.getx() - b.getx()) + (a.gety() - b.gety()) * (a.gety() -
b.gety()));
}

public:
Circle()
{
radius = 0;
}
Circle(Point centr, Point anypoint)
{
centre = centr;
topvertex = anypoint;
radius = calculaterad(centre, anypoint);
}
Circle(Point p, double rad)
{
setradius(rad);
centre = p;
topvertex.setx(p.getx());
```

```
topvertex.sety(p.gety() + int(radius));
}
void setcentre(int x, int y)
{
centre.setx(x);
centre.sety(y);
}
void move(int offsetx, int offsety) override
{
centre.setx(centre.getx() + offsetx);
centre.sety(centre.gety() + offsety);
}
void resize(int k) override
{
int tmp = radius;
radius *= k;
topvertex.sety(topvertex.gety() + radius - tmp);
}

void turn(double a) override
{
}
};

class Fivepointedstar : public Shape
{
public:
//для представления пятиконечной звезды достаточно знать лишь центр и любую другую вершину звезды,для простоты представления будем считать, что нам известна
//верхняя вершина в "каноническом" виде
// 1
//
//5 O 2

// 4 3
// 1 - это верхняя вершина(topvertex), O - это центр
public:
Fivepointedstar()
{
radius = 1;
centre.setx(STANDART_CENTRE);
centre.sety(STANDART_CENTRE);
topvertex.setx(STANDART_CENTRE);
topvertex.sety(STANDART_TOP);
}
Fivepointedstar(Point cen, Point top)
{
centre = cen;
topvertex = top;
calculateradius();
```

```
}
void calculateradius()
{
radius = sqrt((topvertex.getx() - centre.getx()) * (topvertex.getx() - centre.getx()) +
(topvertex.gety() - centre.gety()) * (topvertex.gety() - centre.gety()));
}
void resize(int k) override
{
//по гипотенузе
topvertex.setx(topvertex.getx() * k * k);
topvertex.sety(topvertex.gety() * k * k);
radius *= k;

}
void move(int offsetx,int offsety) override
{
centre.setx(centre.getx() + offsetx);
centre.sety(centre.gety() + offsety);
topvertex.setx(topvertex.getx() + offsetx);
topvertex.sety(topvertex.gety() + offsety);
}

void turn(double a) override
{
int x = topvertex.getx();
int y = topvertex.gety();
topvertex.setx(centre.getx() + (x - centre.getx()) * cos(a) - (y - centre.gety()) * sin(a));
topvertex.sety(centre.gety() + (y - centre.gety()) * cos(a) - (x - centre.getx()) * sin(a));
}
~Fivepointedstar(){}
};

class Sixpointedstar : public Fivepointedstar
{
// 0 * 1
//
// 5 2
//
// 4 3
// * - это topvertex
};

int main(int argc, char const *argv[])
{
general_id = 0;
Circle circle({ 5,6}, 1);
circle.colour.setcolour(RED);
std::cout << circle << std::endl;
circle.resize(5);
circle.turn(30);
```

```
std::cout << circle << std::endl;

Fivepointedstar star({0,1},{5,5});
star.colour.setcolour(GREEN);
std::cout << star << std::endl;

Sixpointedstar sixstar;
std::cout << sixstar << std::endl;

sixstar.colour.setcolour(BLUE);
sixstar.move( -1, 6);
std::cout << sixstar << std::endl;
sixstar.turn(180);
std::cout << sixstar << std::endl;
return 0;
}
```