**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по практической работе №2**

**по дисциплине «ООП»**

Тема: **Наследование**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Кухарев М.А. |
| Преподаватель |  | Терентьев А. О. |

Санкт-Петербург

2018

**Цель работы.**

Изучения понятия наследования. Разработка абстрактного класса. Виртуальные методы и полиморфизм.

**Постановка задачи.**

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 С++.

**Ход работы.**

1. Рассмотрим базовый класс Shape

#pragma once

#include "stdafx.h"

#include "string"

#include "math.h"

using namespace std;

class Shape {

public:

Shape();

virtual ~Shape();

friend ostream&operator<<(ostream&, const Shape&); //поток вывода

protected:

static int ID;

int X = 0, Y = 0;//центр фигуры

int Angle = 0; //угол поворота

double Scale = 1; // изначальный масштаб

string Color = "Белый";//цвет фигуры

virtual void MoveFigure(int, int) = 0; //пермещение

virtual void TurnFigure(int) = 0; //поворот

virtual void ScaleFigure(double) = 0; //масштаб

virtual string GetColorFigure() = 0; //получить цвет

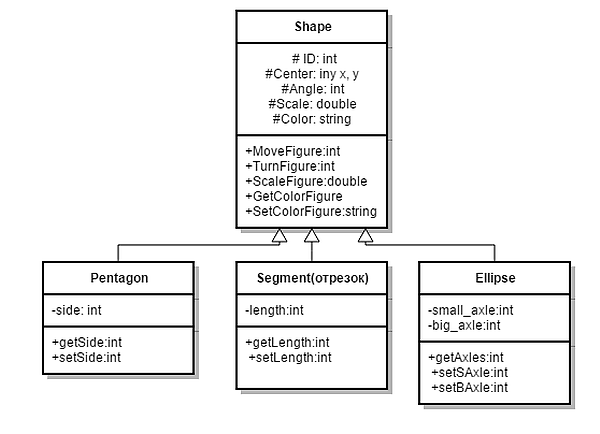
virtual void SetColorFigure(string) = 0; //задать цвет

};

Т.к. в программе будет реализована полиморфная логика, то деструктор объявляется виртуальным. Все объекты будут иметь цвет и размер, так что они объявлены в базовом классе. Так же есть методы установки и получения цвета. Остальные методы являются чистыми виртуальными; их реализация будет своя для каждого из классов, но семантика одна и та же. Приведём её описание:

* virtual void MoveFigure(int, int) = 0; - **перемещение**
* virtual void TurnFigure(int) = 0; - **поворот**
* virtual void ScaleFigure(double) = 0; **- масштаб**
* virtual string GetColorFigure() = 0; **- получить цвет**
* virtual void SetColorFigure(string) = 0**; - задать цвет**

**Наследование:**

****

**Выводы.**

Способность к наследованию встроена в язык C++, что позволяет максимизировать многократное переиспользование кода и даёт возможность производить моделирование предметной области естественным образом. Виртуальные функции дают возможность полиморфической обработки связанных объектов, а абстрактные классы реализацию общего интерфейса.

**ПРИЛОЖЕНИЕ А**

**Shape.h**

#pragma once

#include "stdafx.h"

#include "string"

#include "math.h"

using namespace std;

class Shape {

public:

Shape();

virtual ~Shape();

friend ostream&operator<<(ostream&, const Shape&); //поток вывода

protected:

static int ID;

int X = 0, Y = 0;//центр фигуры

int Angle = 0; //угол поворота

double Scale = 1; // изначальный масштаб

string Color = "Белый";//цвет фигуры

virtual void MoveFigure(int, int) = 0; //пермещение

virtual void TurnFigure(int) = 0; //поворот

virtual void ScaleFigure(double) = 0; //масштаб

virtual string GetColorFigure() = 0; //получить цвет

virtual void SetColorFigure(string) = 0; //задать цвет

};

**Segment.h**

#pragma once

#include "stdafx.h"

#include "Shape.h"

using namespace std;

class Segment : public Shape

{

private:

double Length;

int Id;

public:

Segment(int, int, int, double, int, string);

Segment(int Length);

void SetLength(int);

int GetLength();

friend ostream&operator<<(ostream&a, const Segment&b) {

a << "ID фигуры: id=" << b.Id << "\n";

a << "Центр фигуры: x=" << b.X << "; y=" << b.Y << "\n";

a << "Цвет: " << b.Color << "\n";

a << "Угол поворота фигуры: fi=" << b.Angle << "\n";

a << "Масштаб: delta=" << b.Scale << "\n";

a << "Длина с учетом мастаба: Горизонтальная сторона=" << b.Length\*b.Scale << "\n";

return a;

}

~Segment();

};

**Ellips.h**

#pragma once

#include "stdafx.h"

#include "Shape.h"

class Ellips : public Shape {

public:

int SmallAxle, BigAxle;

int Id;

Ellips(int, int, int, int, double, int, string);

Ellips(int, int);

int GetSAxle();

int GetBAxle();

void SetAxles(int, int);

friend ostream&operator<<(ostream&a, const Ellips&b) {

a << "ID фигуры: id=" << b.Id << "\n";

a << "Центр фигуры: x=" << b.X << "; y=" << b.Y << "\n";

a << "Цвет: " << b.Color << "\n";

a << "Угол поворота фигуры: fi=" << b.Angle << "\n";

a << "Масштаб: delta=" << b.Scale << "\n";

a << "Высота эллипса с учётом масштабирования=" << b.BigAxle\*b.Scale << " Ширина эллипса=" << b.SmallAxle\*b.Scale << "\n";

return a;

}

~Ellips();

};

**Pentagon.h**

#pragma once

#include "stdafx.h"

#include "Shape.h"

class Pentagon : public Shape {

public:

int Side;

int Id;

Pentagon(int, int, int, double, int, string);

Pentagon(int);

int GetSide();

void SetSide(int);

friend ostream&operator<<(ostream&a, const Pentagon&b) {

a << "ID фигуры: id=" << b.Id << "\n";

a << "Центр фигуры: x=" << b.X << "; y=" << b.Y << "\n";

a << "Цвет: " << b.Color << "\n";

a << "Угол поворота фигуры: fi=" << b.Angle << "\n";

a << "Масштаб: delta=" << b.Scale << "\n";

a << "Длины сторон с учетом мастаба: Сторона пятиугольника=" << b.Side\*b.Scale << "\n";

return a;

}

~Pentagon();

};

**ПРИЛОЖЕНИЕ Б**

**Shape.сpp**

#include "stdafx.h"

#include "Shape.h"

int Shape::ID = 0;

Shape::~Shape() {

}

void Shape::MoveFigure(int dx, int dy) {

X = X + dx;

Y = Y + dy;

};

void Shape::TurnFigure(int angle) {

Angle = angle;

}

void Shape::ScaleFigure(double scale) {

Scale = scale;

}

string Shape::GetColorFigure() {

return Color;

}

void Shape::SetColorFigure(string color) {

Color = color;

}

**Segment.сpp**

#include "stdafx.h"

#include "Segment.h"

Segment::Segment(int x, int y, int length, double scale, int angle, string color) {

X = x;

Y = y;

Length = length;

Color = color;

Angle = angle;

Scale = scale;

Id = ID;

ID++;

};

Segment::Segment(int length) {

Length = length;

Id = ID;

ID++;

};

int Segment::GetLength() {

return Length;

}

void Segment::SetLength(int length) {

Length = length;

}

Segment::~Segment() {

};

**Pentagon.сpp**

#include "stdafx.h"

#include "Pentagon.h"

Pentagon::Pentagon(int y, int x, int side, double scale, int angle, string color) {

X = x;

Y = y;

Side = side;

Color = color;

Angle = angle;

Scale = scale;

Id = ID;

ID++;

};

Pentagon::Pentagon(int side) {

Side = side;

Id = ID;

ID++;

};

int Pentagon::GetSide() {

return Side;

}

void Pentagon::SetSide(int side) {

Side = side;

}

Pentagon::~Pentagon() {

};

**Ellips.сpp**

#include "stdafx.h"

#include "Ellips.h"

Ellips::Ellips(int y, int x, int smallAxle, int bigAxle, double scale, int angle, string color) {

X = x;

Y = y;

SmallAxle = smallAxle;

BigAxle = bigAxle;

Color = color;

Angle = angle;

Scale = scale;

Id = ID;

ID++;

};

Ellips::Ellips(int smallAxle, int bigAxle) {

SmallAxle = smallAxle;

BigAxle = bigAxle;

Id = ID;

ID++;

};

int Ellips::GetSAxle() {

return SmallAxle;

}

int Ellips::GetBAxle() {

return BigAxle;

}

void Ellips::SetAxles(int smallAxle, int bigAxle) {

SmallAxle = smallAxle;

BigAxle = bigAxle;

}

Ellips::~Ellips() {

};

**OOP\_Lab2.cpp**

// OOP\_Lab2.cpp: Определяет точку входа консольного приложения

//

#include "stdafx.h"

#include "string"

#include <conio.h>

#include <locale>

#include "Shape.h"

#include "Pentagon.h"

#include "Segment.h"

#include "Ellips.h"

#include "iostream"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

while (true)

{

int choice = 0;

cout << "Выберите действие: " << endl;

cout << "1.Вывести информацию о отрезке" << endl;

cout << "2.Вывести информацию о правильном пятиугольнике" << endl;

cout << "3.Вывести информацию о эллипсе" << endl;

cout << "4.Выход" << endl;

cin >> choice;

switch (choice)

{

case 1:

{

Segment A(0, 0, 3, 2.0, 60, "yellow");

Segment B(1, 1, 3, 3.0, 35, "pink");

cout << A << B;

\_getch();

system("cls");

break;

}

case 2:

{

Pentagon B(0, 0, 3, 4.0, 45, "blue");

cout << B;

\_getch();

system("cls");

break;

}

case 3:

{

Ellips C(0, 3, 3, 4, 3.0, 90, "black");

cout << C;

\_getch();

system("cls");

break;

}

case 4: exit(0);

default:break;

}

}

system("pause");

return 0;

}