Православный Свято-Тихоновский гуманитарный университет

Факультет информатики и прикладной математики

КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа №2

Тема: “Виртуальные методы, наследование, полиморфизм”.

Выполнил: Королев Иван Алексеевич

Проверил: Андрей Сергеевич Евсеев

Группа: 02-1-05

Москва 2021

## 1. Постановка задачи

Дан базовый класс Figure:

class Figure

{

int c; // цвет

bool visible;

protected:

int x,y; // базовая точка

virtual void draw() const;

public:

Figure(int c, int x, int y);

~Figure();

void move(int dx, int dy); // сместить фигуру на (dx,dy) – только видимую

void setBorderColor(int c); // установить цвет фигуры – только видимой

int getBorderColor() const; // получить цвет

void setVisible( bool isVisible = true ); // показать/спрятать фигуру

bool isVisible() const; // признак видимости

virtual void calcParams( float& perimeter, float& area ) const;

// вычислить периметр и площадь фигуры

};

Выполнить:

1. Реализацию методов класса Figure.
2. Скорректировать интерфейс Figure, чтобы методы calcParams и draw стали чисто виртуальными.
3. Определить производный класс Boublik
4. Определить дополнительный метод в производном классе для изменения размеров: void setSizes(радиус1, радиус2);
5. От написанного класса произвести новый дочерний класс - закрашенная фигура.  
   (FilledBoublik ← Boublik ← Figure)
6. Добавить к параметрам конструктора нового дочернего класса цвет заполнения.
7. Определить дополнительный метод у нового дочернего класса для изменения цвета заполнения:

             void setFillColor(int c);

1. Переопределить у нового дочернего класса метод изменения цвета границы:

            void setBorderColor(int c);

           (если новый цвет границы совпадает с текущим цветом заполнения, то цвет границы не изменять)

1. Правильно реализовать деструкторы в родительских и всех производных классах (при необходимости скорректировать интерфейс базового класса).
2. При тестировании объектов классов продемонстрировать полиморфизм (при необходимости скорректировать интерфейс базового класса).
3. При тестировании динамически создать две фигуры 2 разных классов, вызвать все методы.

## 2. Листинг описания класса

Figure.h

class Figure

{

int c; // цвет

bool visible;//состояние существования

protected:

int x, y; // базовая точка

virtual void draw() const = 0; //рисует фигуру

public:

Figure(int c, int x, int y);

virtual ~Figure();

void move(int dx, int dy); // сместить фигуру на (dx,dy) – только видимую

void setBorderColor(int c); // установить цвет фигуры – только видимой

int getBorderColor() const; // получить цвет

void setVisible(bool isVisible = true); // показать/спрятать фигуру

bool isVisible() const; // признак видимости

virtual void calcParams(float& perimeter, float& area) const = 0;

// вычислить периметр и площадь фигуры

};

Boublik.h

#include "Figure.h"

class Boublik : public Figure

{

protected:

float r1;

float r2;

void draw() const;

public:

Boublik(int c, int x, int y, float r1, float r2);

void setSizes(float r1, float r2);

virtual ~Boublik();

void calcParams(float& perimeter, float& area) const; // вычислить периметр и площадь фигуры

};

FilledBoublik.h

#include "Boublik.h"

class FilledBoublik : public Boublik

{

private:

int filled\_c; //цвет закрашенной фигуры 6.

protected:

void draw() const;

public:

FilledBoublik(int filled\_c, int c, int x, int y, float r1, float r2);

virtual ~FilledBoublik();

void setFillcolor(int c); //поменять цвет заливки

void setBorderColor(int c); // изменения цвета границы фигуры

int getFilled\_c() const;

};

## 3. Листинг реализации класса.

Figure.cpp

#include "Figure.h"

#include<iostream>

using std::cout;

using std::endl;

Figure::Figure(int c, int x, int y) { //конструктор

this->visible = true;

this->c = c;

this->x = x;

this->y = y;

}

Figure::~Figure() {}

void Figure::move(int dx, int dy) {

if (isVisible())

{

this->x = this->x + dx;

this->y = this->y + dy;

draw();

}

} // сместить фигуру на (dx,dy) – только видимую

void Figure::setBorderColor(int c) {

if (isVisible())

{

this->c = c;

draw();

}

}// установить цвет фигуры – только видимой

int Figure::getBorderColor() const { return this->c; } // получить цвет

void Figure::setVisible(bool isVisible) {

visible = isVisible;

draw();

} // показать/спрятать фигуру

bool Figure::isVisible() const { return this->visible; } // признак видимости

Boublik.cpp

#include "Boublik.h"

#include <iostream>

Boublik::Boublik(int c, int x, int y, float r1, float r2) : Figure(c, x, y)

{

this->r1 = r1;

this->r2 = r2;

}

void Boublik::setSizes(float r1, float r2) //измменение размера фигуры

{

this->r1 = r1;

this->r2 = r2;

}

void Boublik::calcParams(float& perimeter, float& area) const

{

perimeter = 3.14\*(r2 + r1);

area = 3.14 \* (r2\*r2-r1\*r1);

}

void Boublik::draw() const

{

if (isVisible())

{

std::cout << "Фигура видима!\n"<<"Цвет линии:"<<getBorderColor()<<std::endl;

std::cout << "Центр бублика: ( " << this->x << ", " << this->y << ")\n";

std::cout << "Радиус r1= "<<r1<<" r2= "<<r2<< std::endl;

}

else {

std::cout << "Фигура не видема!\n";

}

}

Boublik::~Boublik()

{

std::cout << "Объект класса Boublik удален!\n";

}

FilledBoublik.cpp

#include "FilledBoublik.h"

#include <iostream>

using namespace std;

FilledBoublik::FilledBoublik(int filled\_c, int c, int x, int y, float r1, float r2): Boublik(c, x, y, r1, r2)

{

this->filled\_c = filled\_c;

}

void FilledBoublik::draw() const

{

if (isVisible())

{

std::cout << "Фигура видима!\n" << "Цвет линии: " << getBorderColor() << std::endl;

std::cout << "Цвет Boublik: " << getFilled\_c()<<endl;

std::cout << "Центр бублика: ( " << this->x << ", " << this->y << ")\n";

std::cout << "Радиус r1= " << r1 << " r2= " << r2 << std::endl;

}

else

{

std::cout << "Фигура не видема!\n";

}

}

FilledBoublik::~FilledBoublik()

{

cout << "Объек класса FilledBoublick\n";

}

void FilledBoublik::setFillcolor(int filled\_c)// изменения цвета заполнения фигуры 7.

{

if (isVisible())

{

this-> filled\_c = filled\_c;

draw();

}

}

void FilledBoublik::setBorderColor(int c) // изменения цвета границы

{

if (c != filled\_c && isVisible())

{

Boublik::setBorderColor(c);

}

}

int FilledBoublik::getFilled\_c() const //вывести цвет закрашенный фтгуры

{

return this->filled\_c;

}

Test.cpp

#include <iostream>

#include "FilledBoublik.h"

using namespace std;

int main() {

setlocale(LC\_ALL, "");

cout << "Создаем объект obj1 без цвета заливки" << endl;

Boublik\* obj1 = new Boublik(12, 0, 0, 5, 7);

Figure\* obj\_figure = obj1; //полиморфизм

cout << endl;

cout << "Показываем obj1:" << endl;

obj\_figure->setVisible();

cout << endl;

cout << "\nСмещаем фигуру на (9; 9):" << endl;

obj\_figure->move(9, 9);

cout << endl;

cout << "Устанавливаем цвет границы фигуры:" << endl;

obj\_figure->setBorderColor(28);

cout << endl;

cout << "Вычисляем периметр и площадь фигуры:" << endl;

float area\_obj1;

float perimeter\_obj1;

obj\_figure->calcParams(perimeter\_obj1, area\_obj1);

cout << "Периметр фигуры: " << perimeter\_obj1 << endl;

cout << "Площадь фигуры: " << area\_obj1 << endl;

delete obj\_figure;

obj1 = nullptr;

cout << endl;

cout << "//////////////////////////////////////////" << endl<< endl;

cout << "Создаем объект obj\_filledBoublik c цветом заливки" << endl;

FilledBoublik\* obj\_filledBoublik = new FilledBoublik(15, 89, 0, 0, 8, 12);

obj\_figure = obj\_filledBoublik; //полиморфизм

cout << "Показываем obj\_filledBoublik:" << endl;

obj\_figure->setVisible();

cout << endl;

cout << "Устанавливаем цвет фигуры:" << endl;

obj\_filledBoublik->setFillcolor(88);

cout << endl;

cout << "Смещаем фигуру на (3;-2):" << endl;

obj\_figure->move(3, -2);

cout << endl;

cout << "Устанавливаем цвет границ фигуры:" << endl;

obj\_figure->setBorderColor(567);

cout << endl;

cout << "Вычисляем периметр и площадь фигуры:" << endl;

float area\_obj\_figure;

float perimeter\_obj\_figure;

obj\_figure->calcParams(perimeter\_obj\_figure, area\_obj\_figure);

cout << "Периметр фигуры: " << perimeter\_obj\_figure << endl;

cout << "Площадь фигуры: " << area\_obj\_figure << endl;

delete obj\_figure;

obj\_figure = NULL;

}

## 4. Результаты работы программы проверки класса (какие действия выполнялись, какие результаты были получены).

1) Создал 2 динамических объекта obj1 типа Boublik и obj\_figure типа Figure

2) Продемонстрировал работу полиморфизма Figure\* obj\_figure = obj1;

3) Вызвал методы:  
   
 obj\_figure->setVisible();  
 obj\_figure->move(9, 9);  
 obj\_figure->setBorderColor(28);  
 obj\_figure->calcParams(perimeter\_obj1, area\_obj1);

4) Удалил объект obj\_figure и obj1 указал на nullptr

Изображение выглядит как текст

Автоматически созданное описание

Рис. 1

5) Создал объект obj\_filledBoublik типа FilledBoublik

6) Вызвал все его методы

7) Удалил объект

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2