Учебный курс: Технологии программирования

Кафедра: ИС Факультет: ИТиП

Лабораторная работа №7. Интерфейсы.

Выполнил: Трофимов В.А. Группа: 1511

Преподаватель: Повышев В.В.

# Задание

*// Реализовать все указанные интерфейсы для классов (согласно варианту):*

*// - D) Прямоугольник*

*// - H) Эллипс (периметр можно считать по любой приближенной формуле:*

*// см. интернет, справочники и т.п.).*

*//*

*// Функционал системы:*

*// - Динамическое добавление фигур пользователем.*

*// - Отобразить все фигуры.*

*// - Общая площадь всех фигур.*

*// - Общий периметр всех фигур.*

*// - Центр масс всей системы.*

*// - Память, занимаемая всеми экземплярами классов.*

*// - Сравнение фигур между собой по массе.*

*// Интерфейс "Геометрическая фигура".*

class GeoFig {

public:

*// Площадь.*

virtual double **square**() = 0;

*// Периметр.*

virtual double **perimeter**() = 0;

};

*// Вектор*

class Vector2D {

public:

double x, y;

};

*// Интерфейс "Физический объект".*

class PhysObject {

public:

*// Масса, кг.*

virtual double **mass**() = 0;

*// Координаты центра масс, м.*

virtual Vector2D **position**() = 0;

*// Сравнение по массе.*

virtual bool operator== ( const PhysObject& ob ) const = 0;

*// Сравнение по массе.*

virtual bool operator< ( const PhysObject& ob ) const = 0;

};

*// Интерфейс "Отображаемый"*

class Printable {

public:

*// Отобразить на экране*

*// (выводить в текстовом виде параметры фигуры).*

virtual void **draw**() = 0;

};

*// Интерфейс для классов, которые можно задать через диалог с пользователем.*

class DialogInitiable {

*// Задать параметры объекта с помощью диалога с пользователем.*

virtual void **initFromDialog**() = 0;

};

*// Интерфейс "Класс"*

class BaseCObject {

public:

*// Имя класса (типа данных).*

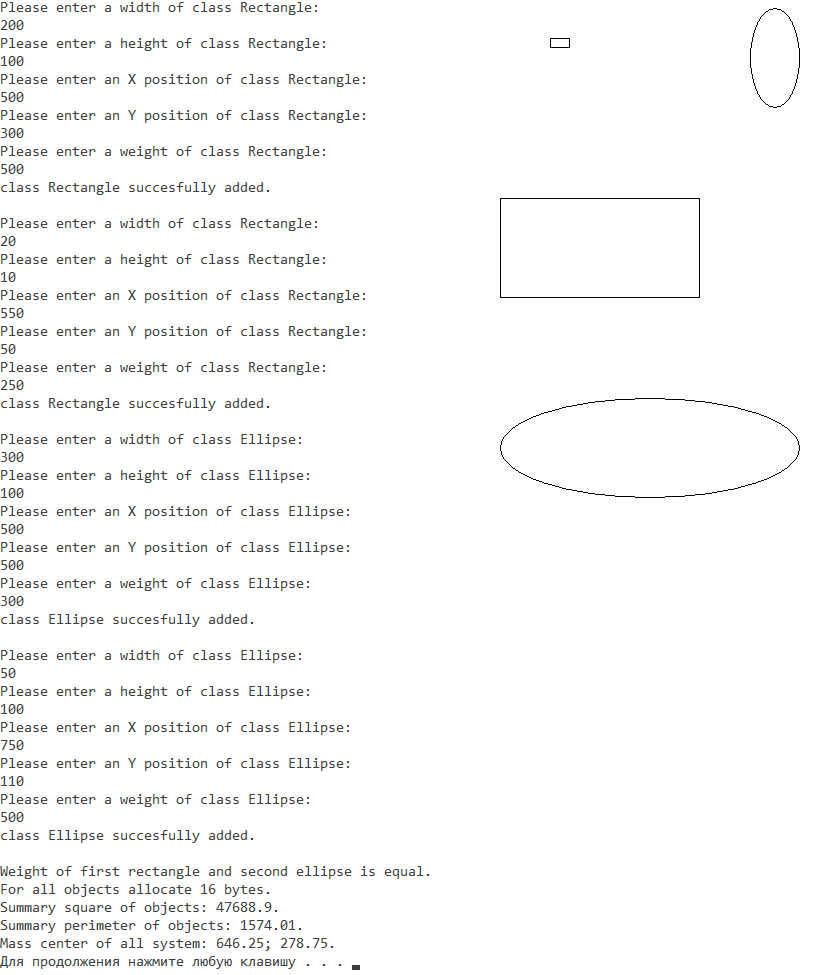
virtual const char\* **classname**() = 0;

*// Размер занимаемой памяти.*

virtual unsigned int **size**() = 0;

};

# Консоль



# Исходный код

#include <typeinfo>

#include <iostream>

#include <string>

#include <list>

const double PI = 3.14159265;

using namespace std;

namespace graphic {

#include <Windows.h>

HDC dc; HWND handle;

};

class Figure {

public:

virtual double square() = 0;

virtual double perimeter() = 0;

};

class Vector2D {

private:

double x, y;

public:

void setX(const double value) {

this -> x = value;

};

void setY(const double value) {

this -> y = value;

}

double getX() const {

return this -> x;

}

double getY() const {

return this -> y;

}

};

class PhysObject {

public:

virtual double mass() const = 0;

virtual Vector2D position() = 0;

virtual bool operator == (const PhysObject & ob) const = 0;

virtual bool operator < (const PhysObject & ob) const = 0;

virtual bool operator > (const PhysObject & ob) const = 0;

virtual bool operator != (const PhysObject & ob) const = 0;

};

class Printable {

public:

virtual void draw() = 0;

};

class DialogInitiable {

virtual void initFromDialog() = 0;

};

class BaseCObject {

public:

virtual const char \* classname() const = 0;

virtual unsigned int size() = 0;

};

class Rectangle : public BaseCObject, public PhysObject, public Figure, public Printable, public DialogInitiable {

protected:

Vector2D pos;

Vector2D sizeXY;

double weight;

public:

//BaseCObject

const char \* classname() const {

return typeid(\*this).name();

}

unsigned int size() {

return sizeof(this);

}

//PhysObject

double mass() const {

return this -> weight;

}

Vector2D position() {

return this -> pos;

}

bool operator == (const PhysObject & ob) const {

return ((this -> weight) == ob.mass());

}

bool operator != (const PhysObject & ob) const {

return ((this -> weight) != ob.mass());

}

bool operator < (const PhysObject & ob) const {

return ((this -> weight) < ob.mass());

}

bool operator > (const PhysObject & ob) const {

return ((this -> weight) > ob.mass());

}

//Figure

double square() {

return ((this -> sizeXY).getX() \* (this -> sizeXY).getY());

}

virtual double perimeter() {

return (2 \* (this -> sizeXY).getX() + 2 \* (this -> sizeXY).getY());

}

//Printable

void draw() {

graphic::handle = graphic::FindWindowA("ConsoleWindowClass", NULL);

graphic::dc = graphic::GetDC(graphic::handle);

graphic::HPEN hPen=(graphic::HPEN)graphic::GetStockObject(WHITE\_PEN);

graphic::HBRUSH hBrush=(graphic::HBRUSH)graphic::GetStockObject(BLACK\_BRUSH);

graphic::HPEN holdpen=(graphic::HPEN)SelectObject(graphic::dc,hPen);

graphic::HBRUSH holdbrush=(graphic::HBRUSH)SelectObject(graphic::dc,hBrush);

graphic::Rectangle(graphic::dc, (int)(this -> pos.getX()), (int)(this -> pos.getY() - this -> sizeXY.getY()),

(int)(this -> pos.getX() + this -> sizeXY.getX()), (int)(this -> pos.getY()));

}

//DialogInitiable

void initFromDialog() {

string temp;

cout << "Please enter a width of " << classname() << ":" << endl;

cin >> temp; (this -> sizeXY).setX(atof(temp.c\_str()));

cout << "Please enter a height of " << classname() << ":" << endl;

cin >> temp; (this -> sizeXY).setY(atof(temp.c\_str()));

cout << "Please enter an X position of " << classname() << ":" << endl;

cin >> temp; (this -> pos).setX(atof(temp.c\_str()));

cout << "Please enter an Y position of " << classname() << ":" << endl;

cin >> temp; (this -> pos).setY(atof(temp.c\_str()));

cout << "Please enter a weight of " << classname() << ":" << endl;

cin >> temp; (this -> weight = atof(temp.c\_str()));

cout << classname() << " succesfully added." << endl << endl;

}

//Custom methods

Rectangle() {

pos.setX(0); pos.setY(0);

sizeXY.setX(0); sizeXY.setY(0);

weight = 0;

}

Vector2D massCenter() {

Vector2D res;

res.setX((this -> sizeXY.getX() / 2) + ((this -> pos).getX()));

res.setY((this -> sizeXY.getY() / 2) + ((this -> pos).getY()));

return res;

}

};

class Ellipse : public Rectangle {

public:

//Figure

double square() {

double a = (this -> sizeXY).getX() / 2;

double b = (this -> sizeXY).getY() / 2;

return PI \* a \* b;

}

virtual double perimeter() {

double a = (this -> sizeXY).getX() / 2;

double b = (this -> sizeXY).getY() / 2;

return 4 \* ((PI \* a \* b + (a - b) \* (a - b)) / (a + b));

}

//Printable

void draw() {

graphic::handle = graphic::FindWindowA("ConsoleWindowClass", NULL);

graphic::dc = graphic::GetDC(graphic::handle);

graphic::HPEN hPen=(graphic::HPEN)graphic::GetStockObject(WHITE\_PEN);

graphic::HBRUSH hBrush=(graphic::HBRUSH)graphic::GetStockObject(BLACK\_BRUSH);

graphic::HPEN holdpen=(graphic::HPEN)SelectObject(graphic::dc,hPen);

graphic::HBRUSH holdbrush=(graphic::HBRUSH)SelectObject(graphic::dc,hBrush);

graphic::Ellipse(graphic::dc, (int)(this -> pos.getX()), (int)(this -> pos.getY() - this -> sizeXY.getY()),

(int)(this -> pos.getX() + this -> sizeXY.getX()), (int)(this -> pos.getY()));

}

//Custom methods

Ellipse() {

pos.setX(0); pos.setY(0);

sizeXY.setX(0); sizeXY.setY(0);

weight = 0;

}

};

class All {

private:

list<Rectangle\*> data;

public:

All() {

}

~All() {

data.clear();

}

void add(Rectangle \* ptr) {

data.push\_back(ptr);

}

void showAll() {

for (list<Rectangle\*>::iterator it = data.begin(); it != data.end(); it++) {

(\*it) -> draw();

}

}

double squareAll() {

double res = 0;

for (list<Rectangle\*>::iterator it = data.begin(); it != data.end(); it++) {

res += ((\*it) -> square());

}

return res;

}

double perimeterAll() {

double res = 0;

for (list<Rectangle\*>::iterator it = data.begin(); it != data.end(); it++) {

res += ((\*it) -> perimeter());

}

return res;

}

int memoryAll() {

int res = 0;

for (list<Rectangle\*>::iterator it = data.begin(); it != data.end(); it++) {

res += ((\*it) -> size());

}

return res;

}

Vector2D massCenterAll() {

Vector2D temp; double resX = 0, resY = 0; int count = 0;

for (list<Rectangle\*>::iterator it = data.begin(); it != data.end(); it++, count++) {

temp = (\*it) -> massCenter(); resX += temp.getX(); resY += temp.getY();

}

temp.setX(resX / count); temp.setY(resY / count);

return temp;

}

};

int main() {

All all;

Rectangle r1; all.add(&r1);

Rectangle r2; all.add(&r2);

Ellipse e1; all.add(&e1);

Ellipse e2; all.add(&e2);

r1.initFromDialog();

r2.initFromDialog();

e1.initFromDialog();

e2.initFromDialog();

all.showAll();

cout << "Weight of first rectangle and second ellipse is " << ((r1 == e2) ? "" : "not") << "equal." <<endl;

cout << "For all objects allocate " << all.memoryAll() << " bytes." << endl;

cout << "Summary square of objects: " << all.squareAll() << "." << endl;

cout << "Summary perimeter of objects: " << all.perimeterAll() << "." << endl;

Vector2D center = all.massCenterAll();

cout << "Mass center of all system: " << center.getX() << "; " << center.getY() << "." << endl;

return 0;

}