Лабараторная работа 4

Клишевич Вадим, M3105

main.cpp

#include <bits/stdc++.h>  
#include "Figures.h"  
using namespace std;  
  
  
int main() {  
 vector<IFigure\*> A;  
 string s;  
 cout << "Write 'add %Figure\_name%' to add that figure\n";  
 cout << "%Figure\_name% should ne 'parallelogram' or 'rectangle'\n";  
 cout << "Write 'showAll' to draw all figures\n";  
 cout << "Write 'sortByMass' to sort objects by mass\n";  
 cout << "Write 'sumSquare' or 'sumPerimeter' to get summary square or perimeter respectively\n";  
 cout << "Write 'sumSize' to get summary size of all objects\n";  
 cout << "Write 'massCenter' to get mass center vector(x, y)\n";  
  
 while (cin >> s) {  
 if (s == "add") {  
 cin >> s;  
 transform(s.begin(), s.end(), s.begin(), [](unsigned char c){ return tolower(c); });  
 if (s == "parallelogram") {  
 A.push\_back(new Parallelogram());  
 } else if (s == "rectangle") {  
 A.push\_back(new Rectangle());  
 } else {  
 cout << "Wrong figure name; you entered '" << s << "'; try again\n";  
 continue;  
 }  
 A.back()->initFromDialogue();  
 } else if (s == "showAll") {  
 for (auto &x : A) {  
 x->draw();  
 }  
 } else if (s == "sumSquare") {  
 double ans = 0;  
 for (auto& x : A) {  
 ans += x->getSquare();  
 }  
 cout << "Summary square = " << ans << "\n";  
 } else if (s == "sumPerimeter") {  
 double ans = 0;  
 for (auto& x : A) {  
 ans += x->getPerimeter();  
 }  
 cout << "Summary perimeter = " << ans << "\n";  
 } else if (s == "sumSize") {  
 double ans = 0;  
 for (auto& x : A) {  
 ans += x->getSize();  
 }  
 cout << "Summary size = " << ans << "\n";  
 } else if (s == "sortByMass") {  
 sort(A.begin(), A.end(), [](IFigure\* x, IFigure\* y) {return \*x < \*y;});  
 cout << "Sorted.\n";  
 } else if (s == "massCenter") {  
 Vector2D center;  
 double massSum = 0;  
 for (auto& x : A) {  
 center.x += x->getPosition().x \* x->getMass();  
 center.y += x->getPosition().y \* x->getMass();  
 massSum += x->getMass();  
 }  
 center.x /= massSum;  
 center.y /= massSum;  
 cout << "Mass center = (" << center.x << "; " << center.y << ")\n";  
 } else {  
 cout << "wrong command; you entered '" << s << "'; try again\n";  
 continue;  
 }  
  
  
 }  
}

Figures.h

//  
// Created by vadim on 7.02.20.  
//  
#include <bits/stdc++.h>  
using namespace std;  
#pragma once  
  
class IGeoFig {  
public:  
 virtual double getSquare() const = 0;  
 virtual double getPerimeter() const = 0;  
};  
  
class Vector2D {  
public:  
 double x, y;  
  
 Vector2D();  
};  
  
class IPhysObject {  
public:  
 virtual double getMass() const = 0;  
 virtual Vector2D getPosition() const = 0;  
 virtual bool operator==(const IPhysObject&) const = 0;  
 virtual bool operator<(const IPhysObject&) const = 0;  
};  
  
class IPrintable {  
public:  
 virtual void draw() const = 0;  
};  
  
class IDialogueInitiable {  
public:  
 virtual void initFromDialogue() = 0;  
};  
  
class IBaseObject {  
public:  
 virtual string getClassName() const = 0;  
 virtual unsigned getSize() const = 0;  
};  
  
  
  
  
class IFigure : public IGeoFig, public IPhysObject, public IPrintable, public IDialogueInitiable, public IBaseObject {  
public:  
  
};  
  
  
  
class Rectangle : public IFigure {  
private:  
 const string name = "Rectangle";  
 double a, b;  
 double mass;  
 Vector2D position{};  
  
public:  
  
 bool operator==(const IPhysObject&) const override;  
  
 bool operator<(const IPhysObject&) const override;  
  
 double getSquare() const override;  
  
 double getPerimeter() const override;  
  
 double getMass() const override;  
  
 Vector2D getPosition() const override;  
  
 void draw() const override;  
  
 void initFromDialogue() override;  
  
 string getClassName() const override;  
  
 unsigned getSize() const override;  
  
 Rectangle();  
  
 Rectangle(const double&, const double&,  
 const double&, const Vector2D&);  
  
 Rectangle(const Rectangle&);  
};  
  
class Parallelogram : public IFigure {  
private:  
 const string name = "Parallelogram";  
 double a, b, alpha;  
 double mass;  
 Vector2D position{};  
  
public:  
  
 bool operator==(const IPhysObject&) const override;  
  
 bool operator<(const IPhysObject&) const override;  
  
 double getSquare() const override;  
  
 double getPerimeter() const override;  
  
 double getMass() const override;  
  
 Vector2D getPosition() const override;  
  
 void draw() const override;  
  
 void initFromDialogue() override;  
  
 string getClassName() const override;  
  
 unsigned getSize() const override;  
  
 Parallelogram();  
  
 Parallelogram(const double&, const double&, const double&,  
 const double&, const Vector2D&);  
  
 Parallelogram(const Parallelogram&);  
};

Figures.cpp

//  
// Created by vadim on 7.02.20.  
//  
#include <bits/stdc++.h>  
#include "Figures.h"  
  
bool Rectangle::operator==(const IPhysObject& x) const {  
 return getMass() == x.getMass();  
}  
  
bool Rectangle::operator<(const IPhysObject& x) const {  
 return getMass() < x.getMass();  
}  
  
double Rectangle::getSquare() const {  
 return a \* b;  
}  
  
double Rectangle::getPerimeter() const {  
 return (a + b) \* 2.;  
}  
  
double Rectangle::getMass() const {  
 return mass;  
}  
  
Vector2D Rectangle::getPosition() const {  
 return position;  
}  
  
void Rectangle::draw() const {  
 cout << "name = " << name << ";\na = " << a << ", b = " << b << ";\n";  
 cout << "mass = " << mass << ";\nsize = " << getSize() << ";\n";  
 cout << "position: x = " << position.x << ", y = " << position.y << ";\n\n";  
}  
  
void Rectangle::initFromDialogue() {  
 double a\_, b\_, mass\_;  
 Vector2D position\_;  
 cout << "Enter Rectangles sides: a, b; Then enter mass and position (x and y)\n";  
 cin >> a\_ >> b\_ >> mass\_ >> position\_.x >> position\_.y;  
 a = a\_; b = b\_;  
 mass = mass\_;  
 position = position\_;  
 cout << "Created successfully.\n";  
}  
  
string Rectangle::getClassName() const {  
 return name;  
}  
  
unsigned Rectangle::getSize() const {  
 return sizeof(\*this);  
}  
  
Rectangle::Rectangle() {  
 a = b = mass = 0;  
 position.x = position.y = 0;  
}  
  
Rectangle::Rectangle(const double& a\_, const double& b\_, const double& mass\_, const Vector2D& position\_) {  
 a = a\_; b = b\_;  
 mass = mass\_;  
 position = position\_;  
}  
  
Rectangle::Rectangle(const Rectangle& x) {  
 a = x.a, b = x.b;  
 mass = x.mass;  
 position = x.position;  
}  
  
  
bool Parallelogram::operator==(const IPhysObject& x) const {  
 return getMass() == x.getMass();  
}  
  
bool Parallelogram::operator<(const IPhysObject& x) const {  
 return getMass() < x.getMass();  
}  
  
double Parallelogram::getSquare() const {  
 return a \* b \* sin(alpha);  
}  
  
double Parallelogram::getPerimeter() const {  
 return (a + b) \* 2.;  
}  
  
double Parallelogram::getMass() const {  
 return mass;  
}  
  
Vector2D Parallelogram::getPosition() const {  
 return position;  
}  
  
void Parallelogram::draw() const {  
 cout << "name = " << name << ";\na = " << a << ", b = " << b << "; angle = " << alpha << "\n";  
 cout << "mass = " << mass << ";\nsize = " << getSize() << ";\n";  
 cout << "position: x = " << position.x << ", y = " << position.y << ";\n\n";  
}  
  
void Parallelogram::initFromDialogue() {  
 double a\_, b\_, alpha\_, mass\_;  
 Vector2D position\_;  
 cout << "Enter Parallelogram sides: a, b and angle (0; Pi); Then enter mass and position (x and y)\n";  
 cin >> a\_ >> b\_ >> alpha\_ >> mass\_ >> position\_.x >> position\_.y;  
 a = a\_; b = b\_;  
 alpha = alpha\_;  
 mass = mass\_;  
 position = position\_;  
 cout << "Created successfully.\n";  
}  
  
string Parallelogram::getClassName() const {  
 return name;  
}  
  
unsigned Parallelogram::getSize() const {  
 return sizeof(\*this);  
}  
  
Parallelogram::Parallelogram() {  
 a = b = mass = 0;  
 alpha = acos(-1.) / 2.;  
 position.x = position.y = 0;  
}  
  
Parallelogram::Parallelogram(const double& a\_, const double& b\_, const double& alpha\_,  
 const double& mass\_, const Vector2D& position\_) {  
 a = a\_; b = b\_;  
 alpha = alpha\_;  
 mass = mass\_;  
 position = position\_;  
}  
  
Parallelogram::Parallelogram(const Parallelogram& x) {  
 a = x.a, b = x.b;  
 alpha = x.alpha;  
 mass = x.mass;  
 position = x.position;  
}  
  
Vector2D::Vector2D() {  
 x = y = 0;  
}