Course: ENSF 614 – Fall 2023

Lab 5:

Instructor: M. Moussavi

Student Name: Emmanuel Alafonye

Submission Date: October 23, 2023.

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include <iostream>

#include "graphicsWorld.h"

using namespace std;

void GraphicsWorld::run() {

// Testing the Point class

Point m(6, 8);

Point n(6, 8);

n.setX(9);

cout << "\nExpected to display the distance between m and n is: 3";

cout << "\nThe distance between m and n is: " << m.distance(n);

cout << "\nExpected second version of the distance function also prints: 3";

cout << "\nThe distance between m and n is again: "

<< Point::distance(m, n);

// Testing the Square class

cout << "\n\nTesting Functions in class Square:" <<endl;

Square s(5, 7, 12, "SQUARE - S");

s.display();

// Testing the Rectangle class

cout << "\nTesting Functions in class Rectangle:" <<endl;

Rectangle a(5, 7, 12, 15, "RECTANGLE A");

a.display();

Rectangle b(16, 7, 8, 9, "RECTANGLE B");

b.display();

double d = a.distance(b);

cout << "\nDistance between rectangle a and b is: " << d <<endl;

Rectangle rec1 = a;

rec1.display();

// Testing assignment operator in class Rectangle

cout << "\nTesting assignment operator in class Rectangle:" <<endl;

Rectangle rec2(3, 4, 11, 7, "RECTANGLE rec2");

rec2.display();

rec2 = a;

a.set\_side\_b(200);

a.set\_side\_a(100);

cout << "\nExpected to display the following values for object rec2: " <<endl;

cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5\n" << "Y-coordinate: 7\n"

<< "Side a: 12\n" << "Side b: 15\n" << "Area: 180\n" << "Perimeter: 54\n";

cout << "\nIf it doesn't, there is a problem with your assignment operator." << std::endl;

rec2.display();

// Testing copy constructor in class Rectangle

cout << "\nTesting copy constructor in class Rectangle:" << std::endl;

Rectangle rec3(a);

rec3.display();

a.set\_side\_b(300);

a.set\_side\_a(400);

cout << "\nExpected to display the following values for object rec3: " <<endl;

cout << "Rectangle Name: RECTANGLE A\n" << "X-coordinate: 5\n" << "Y-coordinate: 7\n"

<< "Side a: 100\n" << "Side b: 200\n" << "Area: 20000\n" << "Perimeter: 600\n";

cout << "\nIf it doesn't, there is a problem with your copy constructor." <<endl;

rec3.display();

// Testing array of pointers and polymorphism

cout << "\nTesting array of pointers and polymorphism:" <<endl;

Shape\* sh[4];

sh[0] = &s;

sh[1] = &b;

sh[2] = &rec1;

sh[3] = &rec3;

for (int i = 0; i < 4; i++) {

sh[i]->display();

}

}

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#pragma once

#include "point.h"

#include "shape.h"

#include "square.h"

#include "rectangle.h"

/\*\*

\* @brief A class representing a graphics world, which manages shapes and their interactions.

\*/

class GraphicsWorld {

public:

void run();

};

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "graphicsWorld.h" // Include the header for your GraphicsWorld class

int main() {

GraphicsWorld program; // Create an instance of your GraphicsWorld class

program.run(); // Call the run method to execute your program

return 0; // Return 0 to indicate successful program execution

}

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "point.h"

#include <iostream>

#include <cmath>

using namespace std;

int Point::pointCount = 0;

/\*\*

\* @brief Constructor to create a Point object with specified coordinates.

\*

\* @param x The X-coordinate of the point.

\* @param y The Y-coordinate of the point.

\*/

Point::Point(double x, double y) : x(x), y(y), id(1000 + pointCount) {

pointCount++;

}

void Point::display() const {

cout << "X-coordinate: " << x <<endl;

cout << "Y-coordinate: " << y <<endl;

}

/\*\*

\* @brief Calculate the Euclidean distance between this point and another point using their coordinates.

\*

\* @param other The other point.

\* @return The Euclidean distance between this point and the other point.

\*/

double Point::distance(const Point& p1, const Point& p2) {

double dx = p1.x - p2.x;

double dy = p1.y - p2.y;

return sqrt(dx \* dx + dy \* dy);

}

double Point::distance(const Point& other) const {

return distance(\*this, other);

}

double Point::getX() const {

return x;

}

double Point::getY() const {

return y;

}

void Point::setX(double x) {

this->x = x;

}

void Point::setY(double y) {

this->y = y;

}

/\*\*

\* @brief Get the total number of Point objects created.

\*

\* @return The total number of Point objects created.

\*/

int Point::counter() {

return pointCount;

}

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#pragma once

/\*\*

\* @brief A class representing a 2D point with X and Y coordinates.

\*/

class Point {

private:

double x;

double y;

static int pointCount;

int id;

public:

/\*\*

\* @brief Constructor to create a Point object.

\*

\* @param x The X-coordinate of the point.

\* @param y The Y-coordinate of the point.

\*/

Point(double x, double y);

void display() const;

static double distance(const Point& p1, const Point& p2);

double distance(const Point& other) const;

double getX() const;

double getY() const;

void setX(double x); // set the X-coordinate

void setY(double y); // set the Y-coordinate

static int counter();

};

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "rectangle.h"

#include <iostream>

using namespace std;

Rectangle::Rectangle(double x, double y, double side\_a, double side\_b, const char\* name)

: Square(x, y, side\_a, name), side\_b(side\_b) {

// Initialize a Rectangle object with provided parameters.

}

double Rectangle::area() const {

return getSideA() \* side\_b;

}

double Rectangle::perimeter() const {

// Calculate and return the perimeter of the rectangle (2 \* (length + width)).

return 2 \* (getSideA() + side\_b);

}

double Rectangle::getSideB() const {

return side\_b;

}

void Rectangle::set\_side\_b(double side) {

side\_b = side;

}

void Rectangle::display() const {

// Display information about the rectangle, including its name, coordinates, side lengths, area, and perimeter.

Square::display();

cout << "Side b: " << side\_b <<endl;

cout << "Area: " << area() <<endl;

cout << "Perimeter: " << perimeter() <<endl;

}

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#pragma once

#include "square.h"

/\*\*

\* @brief A class representing a rectangle, which is a type of quadrilateral shape.

\*

\* This class inherits from the Square class and adds a second side (side\_b) to create a rectangle.

\*/

class Rectangle : public Square {

private:

double side\_b;

public:

/\*\*

\* @brief Constructor to create a Rectangle object.

\*

\* @param x The X-coordinate of the origin.

\* @param y The Y-coordinate of the origin.

\* @param side\_a The length of one side of the rectangle.

\* @param side\_b The length of the second side of the rectangle.

\* @param name The name of the rectangle.

\*/

Rectangle(double x, double y, double side\_a, double side\_b, const char\* name);

double area() const;

double perimeter() const;

double getSideB() const;

void set\_side\_b(double side);

/\*\*

\* @brief Display information about the rectangle, including its name, coordinates, and side lengths.

\*/

void display() const;

};

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "shape.h"

#include <iostream>

#include <cstring>

#include <stdexcept>

using namespace std;

/\*\*

\* @brief A class representing a shape with an origin point and a name.

\*/

Shape::Shape(double x, double y, const char\* name) : origin(x, y), shapeName(nullptr) {

try {

shapeName = new char[strlen(name) + 1];

strcpy(shapeName, name);

} catch (const std::bad\_alloc& e) {

// Handle memory allocation failure

std::cerr << "Memory allocation error: " << e.what() << std::endl;

shapeName = nullptr; // Ensure shapeName is set to nullptr

}

}

Shape::~Shape() {

delete[] shapeName;

}

const Point& Shape::getOrigin() const {

return origin;

}

const char\* Shape::getName() const {

return shapeName;

}

void Shape::display() const {

cout << "Shape Name: " << shapeName << std::endl;

cout << "X-coordinate: " << origin.getX() << std::endl;

cout << "Y-coordinate: " << origin.getY() << std::endl;

}

double Shape::distance(const Shape& s1, const Shape& s2) {

return Point::distance(s1.getOrigin(), s2.getOrigin());

}

double Shape::distance(const Shape& other) const {

return Point::distance(origin, other.getOrigin());

}

/\*\*

\* @brief Move the shape by a specified amount in both the X and Y directions.

\*

\* @param dx The amount to move in the X direction.

\* @param dy The amount to move in the Y direction.

\*/

void Shape::move(double dx, double dy) {

origin = Point(origin.getX() + dx, origin.getY() + dy);

}

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#pragma once

#include "point.h"

/\*\*

\* Represents a geometric shape with an origin point and a name.

\*/

class Shape {

private:

Point origin;

char\* shapeName;

public:

/\*\*

\* Constructor to create a new Shape with the given coordinates and name.

\*

\* @param x The x-coordinate of the origin point.

\* @param y The y-coordinate of the origin point.

\* @param name The name of the shape.

\*/

Shape(double x, double y, const char\* name);

~Shape();

const Point& getOrigin() const;

const char\* getName() const;

void display() const;

/\*\*

\* Calculate the distance between two Shape objects.

\*

\* @param s1 The first Shape for distance calculation.

\* @param s2 The second Shape for distance calculation.

\* @return The distance between s1 and s2 as a double.

\*/

static double distance(const Shape& s1, const Shape& s2);

double distance(const Shape& other) const;

/\*\*

\* Move the shape by specified distances in the x and y directions.

\*

\* @param dx The distance to move the shape in the x-direction.

\* @param dy The distance to move the shape in the y-direction.

\*/

void move(double dx, double dy);

};

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "square.h"

#include <iostream>

using namespace std;

// Constructor for the Square class, initializing the position, side length, and name.

Square::Square(double x, double y, double side\_a, const char\* name) : Shape(x, y, name), side\_a(side\_a) {}

double Square::area() const {

return side\_a \* side\_a;

}

double Square::perimeter() const {

return 4 \* side\_a;

}

double Square::getSideA() const { // Get method to get the side\_a

return side\_a;

}

void Square::set\_side\_a(double side) {

side\_a = side;

}

void Square::display() const {

Shape::display(); // Display method for the base class

cout << "Side a: " << side\_a <<endl;

cout << "Area: " << area() <<endl;

cout << "Perimeter: " << perimeter() <<endl; // Display the perimeter

}

/\*

\* File Name: lab4Exe\_A.cpp

\* Assignment: ENSF 614 Lab 5, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#pragma once

#include "shape.h"

// Defining the Square class that inherit tfrom the shape

class Square : public Shape {

private:

double side\_a;

public:

// The constructor to create the object

Square(double x, double y, double side\_a, const char\* name);

double area() const; // The Area method

double perimeter() const;

double getSideA() const;

void set\_side\_a(double side); // Getter method to retrieve the value of "side\_a."

void display() const; // Used to display and print

};

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Question 5B

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "circle.h"

#include "shape.h"

#include "point.h"

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

Circle::Circle(double x, double y, double radius, const char \*name) : Shape(x, y, name){

setRadius(radius);

}

double Circle::area() const{

return M\_PI \* pow(getRad(), 2);

}

double Circle::perimeter() const{

return 2 \* M\_PI \* getRad();

}

double Circle::getRad() const{

return rad;

}

void Circle::setRadius(double radius){

rad = radius;

}

void Circle::display() const{

getO().display();

cout << "Radius: " << getRad() << endl;

cout << "Area: " << area() << endl;

cout << "Perimeter: " << perimeter() << endl;

}

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "rectangle.h"

#include "shape.h"

#pragma once

/\*\*

\* The Circle class represents a circle with a center point, radius, and name.

\*/

class Circle : virtual public Shape{

protected:

double rad;

public:

/\*\*

\* Constructor to create a Circle object.

\*

\* @param x The x-coordinate of the center.

\* @param y The y-coordinate of the center.

\* @param radius The radius of the circle.

\* @param name The name of the circle.

\*/

Circle(double x, double y, double r, const char \*name);

double area() const;

double perimeter() const;

double getRad() const;

void setRadius(double radius);

void display() const;

};

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "curvecut.h"

#include "circle.h"

#include "shape.h"

#include "point.h"

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

/\*\*

\* Constructor to create a CurveCut object by specifying its position, width, length, cut radius, and name.

\*

\* @param x The x-coordinate of the CurveCut's position.

\* @param y The y-coordinate of the CurveCut's position.

\* @param width The width of the rectangular section.

\* @param length The length of one side of the rectangle.

\* @param cutRadius The radius of the circular cut.

\* @param name The name of the CurveCut.

\*/

CurveCut::CurveCut(double x, double y, double width, double length, double cutRadius, const char \*name)

: Shape(x, y, name), Circle(x, y, cutRadius, name), Rectangle(x, y, width, length, name){

double minLength = width < length ? width : length;

if (cutRadius > minLength)

{

cerr << "\n. Error: Cut radius is too large for the given dimensions.\n";

exit(1);

}

}

double CurveCut::area() const{

return (Rectangle::area() - (Circle::area() / 4));

}

double CurveCut::perimeter() const{

return Rectangle::perimeter() - (2 \* getRad()) + (Circle::perimeter() / 4);

}

/\*\*

\* Display information about the CurveCut, including its name, position, width, length, and cut radius.

\*/

void CurveCut::display() const{

cout << "CurveCut Name: " << getName() << endl;

getO().display();

cout << "Width: " << getSideA() << endl;

cout << "Length: " << getSideB() << endl;

cout << "Radius of the cut: " << getRad() << endl;

}

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "circle.h"

#include "rectangle.h"

#pragma once

/\*\*

\* The CurveCut class represents a shape created by cutting a circular section from a rectangle.

\* It inherits properties from both the Circle and Rectangle classes.

\*/

class CurveCut : public Circle, public Rectangle{

protected:

double width;

public:

/\*\*

\* Constructor to create a CurveCut object.

\*

\* @param x The x-coordinate of the CurveCut's position.

\* @param y The y-coordinate of the CurveCut's position.

\* @param a The length of one side of the rectangle (sideA).

\* @param width The width of the rectangular section.

\* @param radius The radius of the circular section.

\* @param name The name of the CurveCut.

\*/

CurveCut(double x, double y, double a, double width, double radius, const char \*name);

double area() const;

double perimeter() const;

/\*\*

\* Display information about the CurveCut, including its name, position, side length (sideA), width, area, and perimeter.

\*/

void display() const;

};

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "graphicsWorld.h"

#include "curvecut.h"

#include "circle.h"

#include "rectangle.h"

#include "square.h"

#include "shape.h"

#include "point.h"

#include <iostream>

using namespace std;

void GraphicsWorld::run()

{

Point m(6, 8);

Point n(6, 8);

n.setx(9);

cout << "\nExpected to display the distance between m and n is: 3";

cout << "\nThe distance between m and n is: " << m.distance(n);

cout << "\nExpected second version of the distance function also print: 3";

cout << "\nThe distance between m and n is again: " << Point::distance(m, n);

cout << "\n\nTesting Functions in class Square:" << endl;

Square s(5, 7, 12, "SQUARE - S");

s.display();

cout << "\nTesting Functions in class Rectangle:" << endl;

Rectangle a(5, 7, 12, 15, "RECTANGLE A");

a.display();

Rectangle b(16, 7, 8, 9, "RECTANGLE B");

b.display();

double d = a.distance(b);

cout << "\nDistance between square a, and b is: " << d << endl;

Rectangle rec1 = a;

rec1.display();

cout << "\nTesting assignment operator in class Rectangle:" << endl;

Rectangle rec2(3, 4, 11, 7, "RECTANGLE rec2");

rec2.display();

rec2 = a;

a.setSideB(200);

a.setSideA(100);

cout << "\nExpected to display the following values for objec rec2: " << endl;

cout << "Rectangle Name: RECTANGLE A\n"

<< "X-coordinate: 5\n"

<< "Y-coordinate: 7\n"

<< "Side a: 12\n"

<< "Side b: 15\n"

<< "Area: 180\n"

<< "Perimeter: 54\n";

cout << "\nIf it doesn't there is a problem with your assignment operator.\n"

<< endl;

rec2.display();

cout << "\nTesting copy constructor in class Rectangle:" << endl;

Rectangle rec3(a);

rec3.display();

a.setSideB(300);

a.setSideA(400);

cout << "\nExpected to display the following values for objec rec2: " << endl;

cout << "Rectangle Name: RECTANGLE A\n"

<< "X-coordinate: 5\n"

<< "Y-coordinate: 7\n"

<< "Side a: 100\n"

<< "Side b: 200\n"

<< "Area: 20000\n"

<< "Perimeter: 600\n";

cout << "\nIf it doesn't there is a problem with your assignment operator.\n"

<< endl;

rec3.display();

cout << "\nTesting array of pointers and polymorphism:" << endl;

Shape \*sh[4];

sh[0] = &s;

sh[1] = &b;

sh[2] = &rec1;

sh[3] = &rec3;

sh[0]->display();

sh[1]->display();

sh[2]->display();

sh[3]->display();

cout << "\nTesting Functions in class Circle:" << endl;

Circle c(3, 5, 9, "CIRCLE C");

c.display();

cout << "the area of " << c.getName() << " is: " << c.area() << endl;

cout << "the perimeter of " << c.getName() << " is: " << c.perimeter() << endl;

d = a.distance(c);

cout << "\nThe distance between rectangle a and circle c is: " << d << endl;

CurveCut rc(6, 5, 10, 12, 9, "CurveCut rc");

rc.display();

cout << "the area of " << rc.getName() << " is: " << rc.area() << endl;

cout << "the perimeter of " << rc.getName() << " is: " << rc.perimeter();

d = rc.distance(c);

cout << "\nThe distance between rc and c is: " << d << endl;

sh[0] = &s;

sh[1] = &a;

sh[2] = &c;

sh[3] = &rc;

sh[0]->display();

cout << "The area of " << sh[0]->getName() << " is: " << sh[0]->area();

cout << "\nthe perimeter of " << sh[0]->getName() << " is: " << sh[0]->perimeter() << endl << endl;

sh[1]->display();

cout << "\nThe area of " << sh[1]->getName() << " is: " << sh[1]->area();

cout << "\nthe perimeter of " << sh[0]->getName() << " is: " << sh[1]->perimeter() << endl << endl;

sh[2]->display();

cout << "\nThe area of " << sh[2]->getName() << " is: " << sh[2]->area();

cout << "\nthe circumference of " << sh[2]->getName() << " is: " << sh[2]->perimeter() << endl << endl;

sh[3]->display();

cout << "\nThe area of " << sh[3]->getName() << " is: " << sh[3]->area();

cout << "\nthe perimeter of " << sh[3]->getName() << " is: " << sh[3]->perimeter() << endl << endl;

cout << "\nTesting copy constructor in class CurveCut:" << endl;

CurveCut cc = rc;

cc.display();

cout << "\nTesting assignment operator in class CurveCut:" << endl;

CurveCut cc2(2, 5, 100, 12, 9, "CurveCut cc2");

cc2.display();

cc2 = cc;

cc2.display();

}

int main(){

GraphicsWorld functionRun;

functionRun.run();

return 0;

}

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#pragma once

class GraphicsWorld {

public:

void run();

};

/\*

\* File Name: lab4Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "point.h"

#include <stdio.h>

#include <iostream>

#include <math.h>

#include <iomanip>

using namespace std;

int Point::pointCount = 0;

Point::Point(double x, double y){

x = x;

y = y;

id = ++pointCount + 1000;

}

Point::Point(const Point &p2){

x = p2.getx();

y = p2.gety();

id = ++pointCount + 1000;

}

Point &Point::operator=(const Point &h){

if (this != &h){

x = h.getx();

y = h.gety();

id = ++pointCount + 1000;

}

return \*this;

}

Point::~Point(){

--pointCount;

}

void Point::display() const {

cout <<"X-coordinate: " << getx() << endl;

cout <<"Y-coordinate: " << gety() << endl;

}

double Point::getx() const{

return this->x;

}

double Point::gety() const{

return y;

}

void Point::setx(double x){

x = x;

}

void Point::sety(double y){

y = y;

}

int Point::counter() const{

return pointCount;

}

double Point::distance(const Point &p1) const{

double valueX = pow((getx() - p1.getx()), 2);

double valueY = pow((gety() - p1.gety()), 2);

return sqrt(valueX + valueY);

}

double Point::distance(const Point &p1, const Point &p2){

double valueX = pow((p1.getx() - p2.getx()), 2);

double valueY = pow((p1.gety() - p2.gety()), 2);

return sqrt(valueX + valueY);

}

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#pragma once

/\*\*

\* The Point class represents a point in a two-dimensional space.

\*/

class Point{

private:

double x;

double y;

static int pointCount;

int id;

public:

/\*\*

\* Constructor to create a Point object with specified coordinates.

\*

\* @param a The x-coordinate of the point.

\* @param b The y-coordinate of the point.

\*/

Point(double a, double b);

~Point();

Point(const Point &other);

Point &operator=(const Point &rhs);

void display() const;

double getx() const;

double gety() const;

void setx(double a);

void sety(double b);

/\*\*

\* Calculate the distance between this point and another point.

\*

\* @param p3 The other Point object.

\* @return The distance between this point and the other point.

\*/

int counter() const;

double distance(const Point &p3) const;

/\*\*

\* Calculate the distance between two points.

\*

\* @param p1 The first Point object.

\* @param p2 The second Point object.

\* @return The distance between the two points.

\*/

static double distance(const Point &p1, const Point &p2);

};

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "rectangle.h"

#include "square.h"

#include "shape.h"

#include "point.h"

#include <iostream>

#include <iomanip>

using namespace std;

/\*\*

\* The Rectangle class represents a rectangle with two different side lengths, position, and name.

\* It inherits from the Square class and adds a second side length (sideB) to create a rectangle.

\*/

Rectangle::Rectangle(double x, double y, double a, double b, const char \*name)

: Shape(x, y, name), Square(x, y, a, name){

setSideB(b);

}

double Rectangle::area() const{

return (getSideA() \* getSideB());

}

/\*\*

\* Calculate the perimeter of the rectangle.

\*

\* @return The perimeter of the rectangle.

\*/

double Rectangle::perimeter() const{

return (2 \* (getSideA() + getSideB()));

}

double Rectangle::getSideB() const{

return sideB;

}

void Rectangle::setSideB(double side){

sideB = side;

}

/\*\*

\* Display information about the rectangle, including its name, position, side lengths (sideA and sideB), area, and perimeter.

\*/

void Rectangle::display() const{

cout << "Rectangle Name: " << getName() << endl;

getO().display();

cout << "Side A: " << getSideA() << endl;

cout << "Side B: " << getSideB() << endl;

cout << "Area: " << area() << endl;

cout << "Perimeter: " << perimeter() << endl;

}

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "square.h"

#pragma once

/\*\*

\* The Rectangle class represents a rectangle with two different side lengths, position, and name.

\* It inherits from the Square class and adds a second side length (sideB) to create a rectangle.

\*/

class Rectangle : public Square{

protected:

double sideB;

public:

/\*\*

\* Constructor to create a Rectangle object.

\*

\* @param x The x-coordinate of the rectangle's position.

\* @param y The y-coordinate of the rectangle's position.

\* @param a The length of one side of the rectangle (sideA).

\* @param b The length of the second side of the rectangle (sideB).

\* @param sName The name of the rectangle.

\*/

Rectangle(double x, double y, double a, double b, const char \*sName);

double area() const;

double perimeter() const;

/\*\*

\* Get the length of the second side of the rectangle.

\*

\* @return The length of the second side of the rectangle (sideB).

\*/

double getSideB() const;

void setSideB(double side);

/\*\*

\* Display information about the rectangle, including its name, position, side lengths (sideA and sideB), area, and perimeter.

\*/

void display() const;

};

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "shape.h"

#include "point.h"

#include <stdio.h>

#include <string.h>

#include <iostream>

using namespace std;

// Constructor for the Shape class.

Shape::Shape(double x, double y, const char \*sName) : origin(Point(x, y)){

name = new char[strlen(sName) + 1];

strcpy(this->name, sName);

}

// Destructor for the Shape class.

Shape::~Shape(){

delete[] name;

name = nullptr;

}

Shape::Shape(const Shape &s) : origin(Point(s.getO().getx(), s.getO().gety())){

name = new char[strlen(s.getName()) + 1];

strcpy(name, s.getName());

}

Shape &Shape::operator=(const Shape &h){

if (this != &h)

{

delete[] name;

origin = Point(h.getO().getx(), h.getO().gety());

name = new char[strlen(h.getName()) + 1];

strcpy(name, h.getName());

}

return \*this;

}

void Shape::display() const {

cout << "Name : " << getName() << endl;

getO().display();

}

const Point &Shape::getO() const{

return origin;

}

// Get the name of the shape.

const char \*Shape::getName() const{

return name;

}

double Shape::distance(Shape &s1) const{

double dist = getO().distance(s1.getO());

return dist;

}

// Calculate the distance between two shapes using their origins.

double Shape::distance(Shape &s1, Shape &s2){

double dist = s1.getO().distance(s1.getO(), s2.getO());

return dist;

}

// Move the shape by a specified amount in the X and Y directions.

void Shape::move(double moveX, double moveY){

double valueX = getO().getx();

double valueY = getO().gety();

origin.setx(valueX + moveX);

origin.sety(valueY + moveY);

}

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "point.h"

#pragma once

class Shape

{

protected:

Point origin;

char \*name;

public:

Shape(double x, double y, const char \*sName);

virtual ~Shape();

Shape(const Shape &s);

Shape& operator=(const Shape &h);

const Point &getO() const;

const char \*getName() const;

virtual void display() const;

virtual double distance(Shape &S) const;

static double distance(Shape &s1, Shape &s2);

virtual double area() const = 0;

virtual double perimeter() const = 0;

void move (double dx, double dy);

};

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "square.h"

#include "shape.h"

#include "point.h"

#include <iostream>

#include <iomanip>

using namespace std;

/\*\*

\* The Square class represents a square with a side length, position, and name.

\*/

Square::Square(double x, double y, double side, const char \*name) : Shape(x, y, name){

setSideA(side);

}

double Square::area() const{

return getSideA() \* getSideA();

}

double Square::perimeter() const{

return getSideA() \* 4;

}

double Square::getSideA() const{

return sideA;

}

void Square::setSideA(double side){

sideA = side;

}

void Square::display() const{

cout << "Name: " << getName() << endl;

getO().display();

/\*\*

\* Display information about the square, including its name, position, side length, area, and perimeter.

\*/

cout << "Side a: " << getSideA() << endl;

cout << "Area: " << area() << endl;

cout << "Perimeter: " << perimeter() << endl;

}

/\*

\* File Name: lab5Exe\_B.cpp

\* Assignment: ENSF 614 Lab 5, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 23, 2023.

\*/

#include "shape.h"

#include "point.h"

#pragma once

class Square : virtual public Shape{

protected:

double sideA;

public:

// The constructor to create the object

Square(double x, double y, double side, const char \*sName);

double area() const;

double perimeter() const;

double getSideA() const;

void setSideA(double side); // Getter method to retrieve the value of "sideA."

void display() const; // Used to display and print

};

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated