**Names:** Jack Shenfield and Marley Cheema

**Course Name:** Advanced System Analysis and Software Design

**Course Code:** ENSF 614

**Assignment Number:** Lab 03

**Submission Date:** 30/09/2025

Exercise A

Source code:

/\*

\* File Name: point.h

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

#ifndef POINT\_H

#define POINT\_H

// declaration of class point

class point {

private:

double x;

double y;

int id;

static int count; // static so that all points share the same count

public:

point() = delete; // delete default constructor // inspired by chatgpt

point(double a, double b);

// PROMISES: constructs a point object with given x and y coordinates

void display();

// PROMISES: displays a Point object's coordinates and id number

double get\_x() const;

// PROMISES: returns x coordinate of point object

double get\_y() const;

// PROMISES: returns y coordinate of point object

int get\_id() const;

// PROMISES: returns id number of point object

void set\_x(double a);

// PROMISES: sets x coordinate of point object

void set\_y(double b);

// PROMISES: sets y coordinate of point object

static int counter();

// PROMISES: returns number of point objects created

double distance(const point& other);

// PROMISES: returns the distance between two points

static double stat\_distance(const point& a, const point& b);

// PROMISES: static veresion of distance function

};

#endif

/\*

\* File Name: point.cpp

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

// implementation of class point

#include "point.h"

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

point::point(double a, double b){

x = a;

y = b;

id = 1000 + counter();

}

void point::display(){

// printout lines

cout << "X-coordinate: " << fixed << setprecision(2) << x << endl;

cout << "Y-coordinate: " << fixed << setprecision(2) << y << endl;

return;

}

double point::get\_x() const{

return x;

}

double point::get\_y() const{

return y;

}

int point::get\_id() const{

return id;

}

void point::set\_x(double a){

x = a;

return;

}

void point::set\_y(double b){

y = b;

return;

}

int point::count = 0; // static member initialized at 0

int point::counter(){

return ++count; // increment count and then return

}

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

// pythagorean theorem

double xlen = other.get\_x() - x;

double ylen = other.get\_y() - y;

return sqrt((xlen\*xlen) + (ylen\*ylen));

}

double point::stat\_distance(const point& a, const point& b){

// pythagorean theorem

double xlen = b.get\_x() - a.get\_x();

double ylen = b.get\_y() - a.get\_y();

return sqrt((xlen\*xlen) + (ylen\*ylen));

}

/\*

\* File Name: shape.h

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

// declaration of class shape

#ifndef SHAPE\_H

#define SHAPE\_H

#include "point.h"

class shape {

private:

point origin;

char\* shapeName;

public:

shape() = delete; // delete default constructor

shape(point& a, const char\* name);

// PROMISES: constructs a new shape object at point a with a given name of type string

virtual ~shape();

// PROMISES: destructor for shape

void move(double dx, double dy);

// PROMISES: moves the shape by x+dx and y+dy

const point& get\_origin() const;

// PROMISES: returns the origin point of the shape

const char\* get\_name() const;

// PROMISES: returns the name of the shape

double distance(shape& other);

// PROMISES: returns the distances between the origins of two shapes

static double stat\_distance(const shape& a, const shape& b);

// PROMISES: static version of distance function

shape(const shape& other);

// PROMISES: copy constructor for shape

shape& operator=(const shape& other);

// PROMISES: assignment operator for shape

virtual void display() const;

// PROMISES: displays the shape's name and origin

virtual double area() const = 0; // pure virtual

virtual double perimeter() const = 0; // pure virtual

};

#endif

/\*

\* File Name: shape.cpp

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

#include "shape.h"

#include <iostream>

#include <iomanip>

#include <cmath>

#include <cstring>

using namespace std;

// implementation of class shape

shape::shape(point& a, const char\* name): origin(a.get\_x(), a.get\_y()){

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

strcpy(shapeName, name);

return;

}

shape::~shape(){

delete [] shapeName; // free memory

shapeName = nullptr; // avoid pointer pointing at deallocated memory

}

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

// move origin x and y by inputted dx and dy

origin.set\_x(origin.get\_x() + dx);

origin.set\_y(origin.get\_y() + dy);

return;

}

const point& shape::get\_origin() const{

return origin;

}

const char\* shape::get\_name() const{

return shapeName;

}

double shape::distance(shape& other){

return origin.distance(other.get\_origin()); // use previously defined functions

}

double shape::stat\_distance(const shape& a, const shape& b){

return point::stat\_distance(a.get\_origin(), b.get\_origin()); // use previously defined functions

}

shape::shape(const shape& other): origin(other.get\_origin()){

char\* temp = new char[strlen(other.get\_name()) + 1];

strcpy(temp, other.get\_name());

shapeName = temp;

return;

}

shape& shape::operator=(const shape& other){

if(this != &other){ // self assignment check

origin = other.get\_origin(); // use previously defined functions to get values

char\* temp = new char[strlen(other.get\_name()) + 1];

strcpy(temp, other.get\_name());

delete [] shapeName; // free previous name memory

shapeName = temp;

}

return \*this;

}

void shape::display() const{

// printout lines

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

cout << "X-coordinate: " << fixed << setprecision(2) << origin.get\_x() << endl;

cout << "Y-coordinate: " << fixed << setprecision(2) << origin.get\_y() << endl;

return;

}

/\*

\* File Name: square.h

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

#ifndef SQUARE\_H

#define SQUARE\_H

#include "shape.h"

// declaration of class square, derived from shape

class square : public virtual shape {

private:

double side\_a;

public:

square() = delete; // delete default constructor

square(point& origin, const char\* name, double side);

// PROMISES: creates a square object

// REQUIRES: origin point, name is a string, side is double and > 0

double get\_side\_a() const;

// PROMISES: returns side\_a value

void set\_side\_a(double side);

// PROMISES: sets side\_a to inputted side value

// REQUIRES: side is a double and greater than 0

double area() const;

// PROMISES: returns area of the square object

double perimeter() const;

// PROMISES: returns perimeter of the square object

void display() const;

// PROMISES: displays the name, origin, side lenght, perimeter, and area of square object

};

#endif

/\*

\* File Name: square.cpp

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

#include "square.h"

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

// constructor

square::square(point& origin, const char\* name, double side): shape(origin, name), side\_a(side){

// using member initializer list

}

// getter

double square::get\_side\_a() const {

return side\_a;

}

// setter

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

side\_a = side;

}

// area calculation

double square::area() const {

return (side\_a\*side\_a);

}

// perimeter calculation

double square::perimeter() const {

return (4\*side\_a);

}

void square::display() const {

// Similar printout to previous classes.

cout << "Square Name: " << get\_name() << endl;

cout << "X-coordinate: " << fixed << setprecision(2) << get\_origin().get\_x() << endl;

cout << "Y-coordinate: " << fixed << setprecision(2) << get\_origin().get\_y() << endl;

cout << "Side a: " << fixed << setprecision(2) << side\_a << endl;

cout << "Area: " << fixed << setprecision(2) << area() << endl;

cout << "Perimeter: " << fixed << setprecision(2) << perimeter() << endl;

}

/\*

\* File Name: rectangle.h

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

#include "square.h"

// declaration of class rectangle, derived from square

class rectangle : public virtual square {

private:

double side\_b;

public:

rectangle() = delete; // delete default constructor

rectangle(point& origin, const char\* name, double side\_a, double side\_b);

// PROMISES: creates a rectangle object

// REQUIRES: origin point, name is a string, two sides are doubles and > 0

double get\_side\_b() const;

// PROMISES: returns side b length

void set\_side\_b(double side);

// PROMISES: sets sdie b length to double value > 0

double area() const; // override

// PROMISES: returns area of rectangle

double perimeter() const; // override

// PROMISES: returns perimeter of rectangle

void display() const;

// PROMISES: displays the name, origin, side lengths, perimeter, and area of rectangle object

};

#endif

/\*

\* File Name: rectangle.cpp

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

#include "rectangle.h"

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

rectangle::rectangle(point& origin, const char\* name, double side\_a, double side\_b\_in): shape(origin, name), square(origin, name, side\_a), side\_b(side\_b\_in){

// using member initializer list

}

// getter for side b

double rectangle::get\_side\_b() const {

return side\_b;

}

// setter for side b

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

side\_b = (side < 0.0) ? 0.0 : side;

}

// area calculation

double rectangle::area() const {

return (get\_side\_a()\*side\_b);

}

// perimeter calculation

double rectangle::perimeter() const {

return 2.0 \* (get\_side\_a() + side\_b); // 2\*side A and 2\*side B

}

// similar display fucnction to square

void rectangle::display() const {

cout << "Rectangle Name: " << get\_name() << endl;

cout << "X-coordinate: " << fixed << setprecision(2) << get\_origin().get\_x() << endl;

cout << "Y-coordinate: " << fixed << setprecision(2) << get\_origin().get\_y() << endl;

cout << "Side a: " << fixed << setprecision(2) << get\_side\_a() << endl;

cout << "Side b: " << fixed << setprecision(2) << get\_side\_b() << endl;

cout << "Area: " << fixed << setprecision(2) << area() << endl;

cout << "Perimeter: " << fixed << setprecision(2) << perimeter() << endl;

}

/\*

\* File Name: circle.h

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield, Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

#ifndef CIRCLE\_H

#define CIRCLE\_H

#include "shape.h"

// declaration of class circle, derived from shape

class circle : public virtual shape {

private:

double radius;

static constexpr double PI = 3.1415;

public:

circle() = delete; // delete default constructor

circle(point& origin, const char\* name, double radius);

// PROMISES: creates a circle object

// REQUIRES: origin point, name is a string, radius is > 0

double get\_radius() const;

// PROMISES: returns radius length

void set\_radius(double side);

// PROMISES: sets radius to new value (> 0)

double area() const; // override

// PROMISES: returns area of circle

double perimeter() const; // override

// PROMISES: returns perimeter of circle

void display() const;

// PROMISES: displays the name, origin, side lengths, perimeter, and area of circle object

};

#endif

/\*

\* File Name: circle.cpp

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

#include "circle.h"

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

circle::circle(point& origin, const char\* name, double radius\_in): shape(origin, name), radius(radius\_in){

// using member initializer list

}

// getter for radius

double circle::get\_radius() const {

return radius;

}

// setter for radius

void circle::set\_radius(double rad) {

radius = (rad < 0.0) ? 0.0 : rad;

}

// area calculation

double circle::area() const {

return (get\_radius()\*get\_radius()\*PI);

}

// perimeter calculation

double circle::perimeter() const {

return 2.0\*PI\*get\_radius(); // 2\*PI\*R

}

// similar display fucnction to square

void circle::display() const {

cout << "Circle Name: " << get\_name() << endl;

cout << "X-coordinate: " << fixed << setprecision(2) << get\_origin().get\_x() << endl;

cout << "Y-coordinate: " << fixed << setprecision(2) << get\_origin().get\_y() << endl;

cout << "Radius: " << fixed << setprecision(2) << get\_radius() << endl;

cout << "Area: " << fixed << setprecision(2) << area() << endl;

cout << "Perimeter: " << fixed << setprecision(2) << perimeter() << endl;

}

/\*

\* File Name: curvecut.h

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

#ifndef CURVECUT\_H

#define CURVECUT\_H

#include "circle.h"

#include "rectangle.h"

// declaration of class curvecut, derived from circle and rectangle

class curvecut: public circle, public rectangle {

public:

curvecut() = delete; // delete default constructor

curvecut(point& origin, const char\* name, double radius, double side\_a, double side\_b);

// PROMISES: creates a curvecut object

// REQUIRES: origin point, name is a string, radius is > 0, two sides are doubles and > 0

void display() const;

// PROMISES: displays the name, origin, radius, side lengths, perimeter, and area of curvecut object

double area() const; // override

// PROMISES: returns area of curvecut object

double perimeter() const; // override

// PROMISES: returns perimeter of curvecut object

};

#endif

/\*

\* File Name: curvecut.cpp

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

#include "curvecut.h"

#include <iostream>

#include <iomanip>

#include <cmath>

#include <cstdlib>

using namespace std;

curvecut::curvecut(point& origin, const char\* name, double radius, double side\_a, double side\_b): shape(origin, name), square(origin, name, side\_a), circle(origin, name, radius), rectangle(origin, name, side\_a, side\_b){

// using member initializer list

if(radius > side\_a || radius > side\_b){

cout << "Error: radius cannot be greater than width or length." << endl;

exit(1);

}

}

void curvecut::display() const{

cout << "Curvecut Name: " << get\_name() << endl;

cout << "X-coordinate: " << fixed << setprecision(2) << get\_origin().get\_x() << endl;

cout << "Y-coordinate: " << fixed << setprecision(2) << get\_origin().get\_y() << endl;

cout << "Width: " << fixed << setprecision(2) << get\_side\_a() << endl;

cout << "Length: " << fixed << setprecision(2) << get\_side\_b() << endl;

cout << "Radius of the cut: " << fixed << setprecision(2) << get\_radius() << endl;

}

double curvecut::area() const{

return (rectangle::area() - (circle::area()/4.0)); // rectangle area minus 1/4 of the circle's area

}

double curvecut::perimeter() const{

return (rectangle::perimeter() - (2.0\*get\_radius()) + (0.25\*circle::perimeter())); // rectangle perimeter minus 2\*radius plus 1/4 circle perimeter

}

/\*

\* File Name: graphicsWorld.h

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

// declaration of class graphicsWorld

#ifndef GRAPHICSWORLD\_H

#define GRAPHICSWORLD\_H

#include "point.h"

#include "shape.h"

#include "square.h"

#include "rectangle.h"

#include "circle.h"

#include "curvecut.h"

class graphicsWorld {

public:

void run();

// PROMISES: tests new circle and curvecut classes

};

#endif

/\*

\* File Name: graphicsWorld.cpp

\* Assignment: Lab 3 Exercise A

\* Lab Section: B01

\* Completed by: Jack Shenfield & Marley Cheema

\* Submission Date: Sept 30, 2025

\*/

// implementation of class graphicsWorld

#include "graphicsWorld.h"

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

void graphicsWorld::run(){

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ASSUME CODE SEGMENT FOR EXERCISE A IS HERE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#if 1

double d = 0;

// Creating square and rectangle to using in shape array with my curvecut and circle

point Q(0,0);

square s (Q, "Square s", 3);

point Z(3,4);

rectangle a (Z, "Rectangle a", 5, 9);

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

point X(3,5);

circle c (X,"CIRCLE C", 9);

c.display();

cout << "the area of " << c.get\_name() <<" is: "<< c.area() << endl;

cout << "the perimeter of " << c.get\_name() << " is: "<< c.perimeter() << endl;

d = a.distance(c);

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

point Y(6,5);

curvecut rc (Y, "CurveCut rc", 10, 12, 9);

rc.display();

cout << "the area of " << rc.get\_name() <<" is: "<< rc.area();

cout << "the perimeter of " << rc.get\_name() << " is: "<< rc.perimeter();

d = rc.distance(c);

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

// Using array of Shape pointers:

shape\* sh[4];

sh[0] = &s;

sh[1] = &a;

sh[2] = &c;

sh[3] = &rc;

sh[0]->display();

cout << "\nthe area of "<< sh[0]->get\_name() << "is: "<< sh[0] ->area();

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

sh [1]->display();

cout << "\nthe area of "<< sh[1]->get\_name() << "is: "<< sh[1] ->area();

cout << "\nthe perimeter of " << sh[1]->get\_name () << " is: "<< sh[1]->perimeter();

sh [2]->display();

cout << "\nthe area of "<< sh[2]->get\_name() << "is: "<< sh[2] ->area();

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

sh [3]->display();

cout << "\nthe area of "<< sh[3]->get\_name() << "is: "<< sh[3] ->area();

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

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

curvecut cc = rc;

cc.display();

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

point O(2,5);

curvecut cc2(O, "CurveCut cc2", 100, 12, 9);

cc2.display();

cc2 = cc;

cc2.display();

#endif

}; // END OF FUNCTION run

#include "graphicsWorld.h"

int main() {

graphicsWorld gw; // make a graphicsWorld object

gw.run(); // run all the tests inside run()

return 0;

}

Output:

(base) jbs@Jacks-MacBook-Air Lab3\_Exercise\_A % g++ -std=c++11 -Wall -o graphicsWorld \

main.cpp graphicsWorld.cpp point.cpp shape.cpp square.cpp rectangle.cpp circle.cpp curvecut.cpp

(base) jbs@Jacks-MacBook-Air Lab3\_Exercise\_A % ./graphicsWorld

Testing Functions in class Circle:

Circle Name: CIRCLE C

X-coordinate: 3.00

Y-coordinate: 5.00

Radius: 9.00

Area: 254.46

Perimeter: 56.55

the area of CIRCLE C is: 254.46

the perimeter of CIRCLE C is: 56.55

The distance between rectangle a and circle c is: 1.00Error: radius cannot be greater than width or length.

(base) jbs@Jacks-MacBook-Air Lab3\_Exercise\_A %

Exercise C

Source code:

Output: