**Course**: Principals of Software Development – ENSF 409

**Lab 9**

**Instructor**: M. Moshirpour

**Student Name**: Mitchell Sawatzky

**Date Submitted**: March 28, 2016

Exercise B

/Users/Mitchell/Desktop/School/Y2T2 ENSF 409/Labs ENSF/9/Untitled Diagram.png

Output for Part I:

This program has been written by: Mitchell Sawatzky.

Submitted at: 12:00 am, March 27 , 2016

Testing Functions in class Point:

X-coordinate: 6

Y-coordinate: 8

X-coordinate: 9

Y-coordinate: 8

The distance between two points m and n is: 3

Testing Functions in class Square:

Square Name: SQUARE - S

X-coordinate: 5

Y-coordinate: 7

the area of SQUARE - S is: 144

the perimeter of SQUARE - S is: 48

Testing Functions in class Rectangle:

Rectangle Name: RECTANGLE A

X-coordinate: 5

Y-coordinate: 7

Rectangle Name: RECTANGLE B

X-coordinate: 16

Y-coordinate: 7

the area of RECTANGLE A is: 180

the perimeter of RECTANGLE A is: 54

The distance between two rectangles is: 11

Testing copy constructor in class Rectangle:

Rectangle Name: RECTANGLE A

X-coordinate: 5

Y-coordinate: 7

Testing assignment operator in class Rectangle:

Rectangle Name: RECTANGLE A

X-coordinate: 5

Y-coordinate: 7

Testing Functions in class Circle:

Circle Name: CIRCLE C

X-coordinate: 3

Y-coordinate: 5

the area of CIRCLE C is: 254.47

the perimeter of CIRCLE C is: 56.5488

The distance between rectangle a and circle c is: 2.82843

Output for Part II:

This program has been written by: Mitchell Sawatzky.

Submitted at: 12:00 am, March 27 , 2016

Testing Functions in class Point:

X-coordinate: 6

Y-coordinate: 8

X-coordinate: 9

Y-coordinate: 8

The distance between two points m and n is: 3

Testing Functions in class Square:

Square Name: SQUARE - S

X-coordinate: 5

Y-coordinate: 7

the area of SQUARE - S is: 144

the perimeter of SQUARE - S is: 48

Testing Functions in class Rectangle:

Rectangle Name: RECTANGLE A

X-coordinate: 5

Y-coordinate: 7

Rectangle Name: RECTANGLE B

X-coordinate: 16

Y-coordinate: 7

the area of RECTANGLE A is: 180

the perimeter of RECTANGLE A is: 54

The distance between two rectangles is: 11

Testing copy constructor in class Rectangle:

Rectangle Name: RECTANGLE A

X-coordinate: 5

Y-coordinate: 7

Testing assignment operator in class Rectangle:

Rectangle Name: RECTANGLE A

X-coordinate: 5

Y-coordinate: 7

Testing Functions in class Circle:

Circle Name: CIRCLE C

X-coordinate: 3

Y-coordinate: 5

the area of CIRCLE C is: 254.47

the perimeter of CIRCLE C is: 56.5488

The distance between rectangle a and circle c is: 2.82843 CornerCut Name: CornerCut rc

X-coordinate: 6

Y-coordinate: 5

Width: 10

Length: 12

Radius of the cut: 9

the area of CornerCut rc is: 56.3826the perimeter of CornerCut rc is: 40.1372

The distance between rc and c is: 3 Shape Name: SQUARE - S

X-coordinate: 5

Y-coordinate: 7

the area of SQUARE - Sis: 144

the perimeter of SQUARE - S is: 48 Shape Name: RECTANGLE A

X-coordinate: 5

Y-coordinate: 7

the area of RECTANGLE Ais: 180

the perimeter of SQUARE - S is: 54 Shape Name: CIRCLE C

X-coordinate: 3

Y-coordinate: 5

the area of CIRCLE Cis: 254.47

the circumference of CIRCLE C is: 56.5488 Shape Name: CornerCut rc

X-coordinate: 6

Y-coordinate: 5

the area of CornerCut rcis: 56.3826

the perimeter of CornerCut rc is: 40.1372

Testing copy constructor in class CornerCut:

CornerCut Name: CornerCut rc

X-coordinate: 6

Y-coordinate: 5

Width: 10

Length: 12

Radius of the cut: 9

Testing assignment operator in class CornerCut:

CornerCut Name: CornerCut cc2

X-coordinate: 2

Y-coordinate: 5

Width: 12

Length: 100

Radius of the cut: 9

CornerCut Name: CornerCut rc

X-coordinate: 6

Y-coordinate: 5

Width: 10

Length: 12

Radius of the cut: 9

circle.cpp

// File: circle.cpp

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#include "circle.h"

#include <iostream>

using namespace std;

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

: Shape(x, y, name) {

this->radius = radius;

}

double Circle::area () {

return M\_PI \* radius \* radius;

}

double Circle::perimeter () {

return 2 \* M\_PI \* radius;

}

double Circle::getRadius () {

return radius;

}

void Circle::setRadius (double radius) {

this->radius = radius;

}

void Circle::display () {

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

getOrigin().display();

}

circle.h

// File: circle.h

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#ifndef CIRCLE\_H

#define CIRCLE\_H

#include "shape.h"

const double M\_PI = 3.1416;

class Circle: virtual public Shape {

public:

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

// PROMISES: Calculate the area of the shape and return it

double area ();

// PROMISES: Calculate the area of the shape and return it

double perimeter ();

// PROMISES: Gets the radius and returns it

double getRadius ();

// PROMISES: Sets the radius to the new value

void setRadius (double radius);

// PROMISES: Outputs shape data to stdout

void display ();

private:

double radius;

};

#endif

cornerCut.cpp

// File: cornerCut.cpp

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#include "cornerCut.h"

#include <iostream>

using namespace std;

CornerCut::CornerCut (double x, double y, double sideA, double sideB, double radius, const char\* name)

: Circle(x, y, radius, name), Rectangle(x, y, sideA, sideB, name), Shape(x, y, name) {

if (radius > sideA || radius > sideB) {

cerr << "Error: Cannot create a CornerCut object with radius larger than width" << endl;

exit(1);

}

}

double CornerCut::area () {

return Rectangle::area() - (0.25 \* Circle::area());

}

double CornerCut::perimeter () {

return Rectangle::perimeter() - (2 \* getRadius()) + (0.25 \* Circle::perimeter());

}

void CornerCut::display () {

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

Circle::getOrigin().display();

if (getSideA() >= getSideB()) {

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

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

} else {

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

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

}

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

}

cornerCut.h

// File: cornerCut.h

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#ifndef CORNERCUT\_H

#define CORNERCUT\_H

#include "circle.h"

#include "rectangle.h"

class CornerCut : public Circle, public Rectangle {

public:

CornerCut (double x, double y, double sideA, double sideB, double radius, const char\* name);

// PROMISES: Calculates the area of the shape

double area ();

// PROMISES: Calculates the perimeter of the shape

double perimeter ();

// PROMISES: Outputs the shape data to stdout

void display ();

};

#endif

graphicWorld.cpp

// File: graphicsWorld.cpp

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#include "rectangle.h"

#include "square.h"

#include "circle.h"

#include "cornerCut.h"

#include <iostream>

using namespace std;

class GraphicsWorld {

public:

void run() {

//=================================== PART 1 =============================================

cout << "\nThis program has been written by: Mitchell Sawatzky." ;

cout << "\nSubmitted at: 12:00 am, March 27 , 2016\n";

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

Point m (6, 8);

Point n (6,8);

n.setX(9);

m.display();

n.display();

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

// Testing the second version of the function distance.

// Put the necessary code in this place

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

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

s.display();

cout << "the area of " << s.getName() <<" is: "<< s.area() << "\n";

cout << "the perimeter of " << s.getName() <<" is: " << s.perimeter() << "\n";

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();

cout << "the area of " << a.getName() <<" is: "<< a.area() << "\n";

cout << "the perimeter of " << a.getName() <<" is: "<< a.perimeter() << "\n";

double d = a.distance(b);

cout << "\nThe distance between two rectangles is: " <<d;

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

Rectangle rec1 = a;

rec1.display();

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

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

rec2 = a;

rec2.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;

// =================================== PART 2 =============================================

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

rc.display();

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

cout << "the perimeter of " << rc.getName() << " 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]->getName() << "is: " << dynamic\_cast<Square\*>(sh[0])->area();

cout << "\nthe perimeter of " << sh[0]->getName() << " is: " << dynamic\_cast<Square\*>(sh[0])->perimeter();

sh[1]->display();

cout << "\nthe area of "<< sh[1]->getName() << "is: " << dynamic\_cast<Rectangle\*>(sh[1])->area();

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

sh[2]->display();

cout << "\nthe area of " << sh[2]->getName() << "is: " << dynamic\_cast<Circle\*>(sh[2])->area();

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

sh[3]->display();

cout << "\nthe area of " << sh[3]->getName() << "is: " << dynamic\_cast<CornerCut\*>(sh[3])->area();

cout << "\nthe perimeter of " << sh[3]->getName() << " is: " << dynamic\_cast<CornerCut\*>(sh[3])->perimeter();

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

CornerCut cc = rc;

cc.display();

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

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

cc2.display();

cc2 = cc;

cc2.display();

}

};

int main () {

GraphicsWorld gw;

gw.run();

return 0;

}

point.cpp

// File: point.cpp

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#include "point.h"

#include <iostream>

#include <cmath>

using namespace std;

int Point::idIndex = 1001;

int Point::objectCount = 0;

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

this->x = x;

this->y = y;

this->id = Point::idIndex;

Point::objectCount++;

Point::idIndex++;

}

Point::~Point () {

Point::objectCount--;

}

void Point::display () const {

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

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

}

double Point::getX () const {

return this->x;

}

double Point::getY () const {

return this->y;

}

double Point::getID () const {

return this->id;

}

void Point::setX (double x) {

this->x = x;

}

void Point::setY (double y) {

this->y = y;

}

int Point::counter () {

return Point::objectCount;

}

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

return sqrt(pow(other.getX() - getX(), 2) + pow(other.getY() - getY(), 2));

}

double Point::distance (const Point& a, const Point& b) {

return sqrt(pow(b.getX() - a.getX(), 2) + pow(b.getY() - a.getY(), 2));

}

point.h

// File: point.h

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#ifndef POINT\_H

#define POINT\_H

class Point {

public:

Point (double x, double y);

~Point ();

// PROMISES: Outputs point data to stdout

void display () const;

// PROMISES: Gets the x coordinate

double getX () const;

// PROMISES: Gets the Y coordinate

double getY () const;

// PROMISES: Gets the ID of the object

double getID () const;

// PROMISES: Sets the x coordinate to the new value

void setX (double x);

// PROMISES: Sets the y coordinate to the new value

void setY (double y);

// PROMISES: Calculates the distance between this object and another

double distance (const Point& other) const;

// PROMISES: Calculates the distance between two objects

static double distance (const Point& a, const Point& b);

// PROMISES: Returns the total number of point objects in memory

static int counter ();

private:

double x;

double y;

int id;

static int idIndex;

static int objectCount;

};

#endif

rectangle.cpp

// File: rectangle.cpp

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#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), Shape(x, y, name) {

this->side\_b = side\_b;

}

double Rectangle::area () {

return side\_b \* getSideA();

}

double Rectangle::perimeter () {

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

}

double Rectangle::getSideB () {

return side\_b;

}

void Rectangle::setSideB (double b) {

side\_b = b;

}

void Rectangle::display () {

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

getOrigin().display();

}

rectangle.h

// File: rectangle.h

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

#include "square.h"

class Rectangle: public Square {

public:

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

// PROMISES: Calculates the area of the shape

double area ();

// PROMISES: Calculates the perimeter of the shape

double perimeter ();

// PROMISES: Gets the side\_b of the shaep

double getSideB ();

// PROMISES: Sets side\_b to the new value

void setSideB (double b);

// PROMISES: Outputs the shape data to stdout

void display ();

private:

double side\_b;

};

#endif

shape.cpp

// File: shape.cpp

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#include "shape.h"

#include "point.h"

#include <iostream>

using namespace std;

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

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

strcpy(shapeName, name);

}

Shape::Shape (const Shape& src) : origin(src.getOrigin().getX(), src.getOrigin().getY()) {

shapeName = new char[strlen(src.getName()) + 1];

strcpy(shapeName, src.getName());

}

Shape::~Shape () {

delete [] shapeName;

}

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

if (this != &rhs) {

if (shapeName) {

delete [] shapeName;

}

origin = Point(rhs.getOrigin().getX(), rhs.getOrigin().getY());

shapeName = new char[strlen(rhs.getName()) + 1];

strcpy(shapeName, rhs.getName());

}

return \*this;

}

const Point& Shape::getOrigin () const {

return origin;

}

char\* Shape::getName () const {

return shapeName;

}

void Shape::display () const {

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

origin.display();

}

double Shape::distance (Shape& other) {

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

}

double Shape::distance (Shape& the\_shape, Shape& other) {

return Point::distance(the\_shape.getOrigin(), other.getOrigin());

}

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

origin.setX(origin.getX() + dx);

origin.setY(origin.getY() + dy);

}

shape.h

// File: shape.h

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#ifndef SHAPE\_H

#define SHAPE\_H

#include "point.h"

class Shape {

public:

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

Shape (const Shape& src);

~Shape ();

Shape& operator= (const Shape& rhs);

// PROMISES: Gets the origin point

const Point& getOrigin () const;

// PROMISES: Gets the name of the shape

char\* getName () const;

// PROMISES: Outputs shape data to stdout

virtual void display () const;

// PROMISES: Calculates the distance between the shape and another shape

double distance (Shape& other);

// PROMISES: Calculates the distance between the shape and another shape

static double distance (Shape& the\_shape, Shape& other);

// PROMISES: Moves the shape by dx and dy

void move (double dx, double dy);

private:

Point origin;

char\* shapeName;

};

#endif

square.cpp

// File: square.cpp

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#include "square.h"

#include <iostream>

using namespace std;

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

: Shape(x, y, name) {

side\_a = side;

}

double Square::area () {

return side\_a \* side\_a;

}

double Square::perimeter () {

return 4 \* side\_a;

}

double Square::getSideA () {

return side\_a;

}

void Square::setSideA (double side) {

side\_a = side;

}

void Square::display () {

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

getOrigin().display();

}

square.h

// File: square.h

// Author: Mitchell Sawatzky

// Date: March 27, 2016

// Class: ENSF 409

#ifndef SQUARE\_H

#define SQUARE\_H

#include "shape.h"

class Square: virtual public Shape {

public:

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

// PROMISES: Calculates the area of the shape

virtual double area ();

// PROMISES: Calculates the perimeter of the shape

virtual double perimeter ();

// PROMISES: Gets the side\_a of the shape

double getSideA ();

// PROMISES: Sets the side\_a of the shape to the new value

void setSideA (double side);

// PROMISES: Outputs the shape data to stdout

virtual void display ();

private:

double side\_a;

};

#endif