Homework #1

Benjamin Lattanzio

Benjamin.Lattanzio@cix.csi.cuny.edu

#pragma once

class Point2D {

private:

int x;

int y;

public:

Point2D() :x(0), y(0) {};//default constructor

Point2D(int, int);

int getPointX()const;

int getPointY()const;

bool equal(Point2D);

void setPointX(int);

void setPointY(int);

void print();

};

#include "Point2D.h"

#include <iostream>

using namespace std;

Point2D::Point2D(int inX, int inY) { x = inX; y = inY; }//constructor initializer

int Point2D::getPointX()const {

return x;

}

int Point2D::getPointY() const {

return y;

}

void Point2D::setPointX(int inX) {

x = inX;

}

void Point2D::setPointY(int inY) {

y = inY;

}

bool Point2D::equal(Point2D inPoint) {//checking if the two points are equal to another

bool check = false;

if (x == inPoint.getPointX() && y == inPoint.getPointY())

check = true;

return check;

}

void Point2D::print() {//printing out the point

cout << "The point is : (" << x << ',' << y << ')' << endl;

}

#pragma once

#include "Point2D.h"

class Circle : public Point2D{

private:

double radius;

public:

Circle() :radius(0) {};//default constructor

Circle(int,int,double) ;

double getRadius()const;

void setRadius(double);

double area();

bool equal(Circle);

void print();

double distance();

double distance(Circle);

};

#include "Circle.h"

#include "Point2D.h"

#include<cmath>

#include <iostream>

using namespace std;

Circle::Circle(int inX, int inY, double inRadius):Point2D(inX,inY) { radius = inRadius; }; //constructor initializer

double Circle::getRadius()const {

return radius;

}

void Circle::setRadius(double inRadius) {

radius = inRadius;

}

double Circle::area() {//finding area of circle

double area = 0;

double pi = 3.14;

return area = pi \* (radius\*radius);

}

bool Circle::equal(Circle inCircle) {//checking if the radius of two circles are equal

bool check = false;

if (radius == inCircle.getRadius())

check = true;

return check;

}

double Circle::distance() {//finding distance from origin to circle

double distance = 0;

distance = sqrt((getPointX()\*getPointX()) + (getPointY()\*getPointY()));

return distance;

}

double Circle::distance(Circle inCircle) {//finding distance between two circles

double distance = 0;

distance = sqrt(((inCircle.getPointX() - getPointX())\*(inCircle.getPointX() - getPointX())) + ((inCircle.getPointY()-getPointY())\*(inCircle.getPointY() - getPointY())));

return distance;

}

void Circle::print() {//printing out radius and area

cout << "The radius of the circle is: " << radius << endl;

cout << "The area of the circle is: " << area() << endl;

}

#pragma once

#include "Circle.h"

#include "Cylinder.h"

class Cylinder: public Circle{

private:

double height;

public:

Cylinder() :height(0) {};//default constructor

Cylinder(int,int,double, double);

double area();

double getHeight()const;

double volume();

void setHeight(double);

bool equal(Cylinder);

void print();

};

#include "Cylinder.h"

#include"Point2D.h"

#include "Circle.h"

#include <iostream>

using namespace std;

Cylinder::Cylinder(int inX, int inY, double radius, double inHeight):Circle (inX,inY,radius) { height = inHeight; };//constructor initializer

double Cylinder::getHeight()const {

return height;

}

void Cylinder::setHeight(double inHeight) {

height = inHeight;

}

double Cylinder::area() {//finding area of cylinder

double pi = 3.14;

double area = 0;

return area = (2 \* pi\*getRadius()\*height) + (2 \* pi\*(getRadius()\*getRadius()));

}

double Cylinder::volume() {//finding volume of cylinder

double volume=0;

double pi = 3.14;

return volume = pi \* (getRadius()\*getRadius())\*height;

}

bool Cylinder::equal(Cylinder inCylinder) {// checking if height and radius are equal between two cylinders

bool check = false;

if (height == inCylinder.getHeight() && getRadius() == inCylinder.getRadius())

check = true;

return check;

}

void Cylinder::print() {//printing out height area and volume

cout << "The height of the cylinder is: " << height << endl;

cout << "The area of the cylinder is: " << area() << endl;

cout << "The volume of the cylinder is: " << volume() << endl;

}

#include <iostream>

#include "Circle.h"

#include"Point2D.h"

#include"Cylinder.h"

using namespace std;

double distance(Circle inCircle);//prototypes for functions

double distance(Circle inOne, Circle inTwo);

int main() {

double distanceOne = 0;

double distanceTwo = 0;

Point2D\* pointOne = new Point2D(3,3);//3 dynamic objects

Circle\* circleOne = new Circle(4,2,7);

Cylinder\* cylinderOne = new Cylinder(2,3,4,6);

Point2D pointTwo(5, 2);//point and all methods

Point2D pointThree(7, 3);

pointTwo.setPointX(3);

pointTwo.setPointY(5);

pointTwo.equal(pointThree);

pointTwo.print();

Circle circleTwo(1,2,2.5);//circle and all methods

Circle circleThree(5, 3, 7);

circleTwo.setRadius(8);

circleTwo.equal(circleThree);

circleTwo.area();

circleTwo.distance();

circleTwo.distance(circleThree);

distanceOne = distance(circleTwo);

distanceTwo = distance(circleTwo, circleThree);

circleTwo.print();

Cylinder cylinderTwo(4,3,2,15);//cylinder and all methods

Cylinder cylinderThree(4,3,2,3);

cylinderTwo.setHeight(10);

cylinderTwo.equal(cylinderThree);

cylinderTwo.area();

cylinderTwo.volume();

cylinderTwo.print();

Circle \* pointerCircleArray[10];//three arrays of object pointers

Point2D \* pointerPoint[10];

Cylinder \* pointerCylinder[10];

pointerCircleArray[0] = &circleTwo;// referencing objects

pointerPoint[0] = &pointTwo;

pointerCylinder[0] = &cylinderTwo;

Circle circleArray[10];//three arrays of objects

Point2D pointArray[10];

Cylinder cylinderArray[10];

system("pause");

return 0;

}

double distance(Circle inCircle) {//non memeber-finding distance from origin to circle

double distance = 0;

distance = sqrt((inCircle.getPointX()\*inCircle.getPointX()) + (inCircle.getPointY()\*inCircle.getPointY()));

return distance;

}

double distance(Circle inOne, Circle inTwo) {//non member-finding distance between two circles

double distance = 0;

distance = sqrt(((inTwo.getPointX() - inOne.getPointX())\*(inTwo.getPointX() - inOne.getPointX())) + ((inTwo.getPointY() - inOne.getPointY())\*(inTwo.getPointY() - inOne.getPointY())));

return distance;

}



