ZOHA AHMED

23I-2555

QUESTION#1:

#include <iostream>

#include <cmath>

using namespace std;

const double PI = 3.14;

class Shape {

public:

    double calculateArea(double radius) {

        return PI \* radius \* radius;

    }

    double calculatePerimeter(double radius) {

        return 2 \* PI \* radius;

    }

    double Area(double length, double width) {

        return length \* width;

    }

    double calculatePerimeter(double length, double width) {

        return 2 \* (length + width);

    }

    double calculateArea(double base, double height) {

        return 0.5 \* base \* height;

    }

    double calculatePerimeter(double s1, double s2, double s3) {

        return s1 + s2 + s3;

    }

};

int main() {

    Shape s;

    double circleRadius = 10.0;

    cout << "Circle with radius " << circleRadius << endl;

    cout << "Area: " << s.calculateArea(circleRadius) << endl;

    cout << "Perimeter: " << s.calculatePerimeter(circleRadius) << endl;

    double rectangleLength = 4.0;

    double rectangleWidth = 6.0;

    cout << "Rectangle with length " << rectangleLength << " and width " << rectangleWidth << endl;

    cout << "Area: " << s.calculateArea(rectangleLength, rectangleWidth) << endl;

    cout << "Perimeter: " << s.calculatePerimeter(rectangleLength, rectangleWidth) << endl;

    double triangleBase = 3.0;

    double triangleHeight = 4.0;

    double triangleS1 = 5.0;

    double triangleS2 = 6.0;

    double triangleS3 = 7.0;

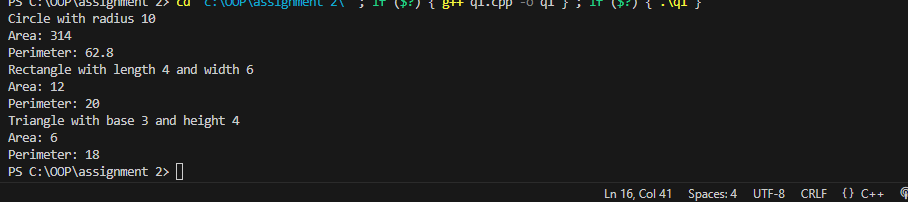
    cout << "Triangle with base " << triangleBase << " and height " << triangleHeight << endl;

    cout << "Area: " << s.calculateArea(triangleBase, triangleHeight) << endl;

    cout << "Perimeter: " << s.calculatePerimeter(triangleS1, triangleS2, triangleS3) << endl;

    return 0;

}



Question# 2:

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

const double PI = 3.14;

class Shape {

public:

    virtual double area() const = 0;

    virtual double perimeter() const = 0;

    virtual void displayProperties() const = 0;

    virtual ~Shape() {}

};

class Circle : public Shape {

private:

    double radius;

public:

    Circle(double r) : radius(r) {}

    double area() const override {

        return M\_PI \* radius \* radius;

    }

    double perimeter() const override {

        return 2 \* M\_PI \* radius;

    }

    void displayProperties() const override {

        cout << "Properties of the Circle:" << endl;

        cout << "Area: " << fixed << setprecision(4) << area() << endl;

        cout << "Perimeter: " << fixed << setprecision(4) << perimeter() << endl;

        cout << "Diameter: " << 2 \* radius << endl;

    }

};

class Rectangle : public Shape {

private:

    double length;

    double width;

public:

    Rectangle(double l, double w) : length(l), width(w) {}

    double area() const override {

        return length \* width;

    }

    double perimeter() const override {

        return 2 \* (length + width);

    }

    void displayProperties() const override {

        cout << "Properties of the Rectangle:" << endl;

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

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

        cout << "Diagonal: " << sqrt(length \* length + width \* width) << endl;

    }

};

class Square : public Rectangle {

public:

    Square(double side) : Rectangle(side, side) {}

    void displayProperties() const override {

        cout << "Properties of the Square:" << endl;

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

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

        cout << "Diagonal: " << sqrt(2) \* Rectangle::area() << endl;

    }

};

class Triangle : public Shape {

private:

    double side1, side2, side3;

public:

    Triangle(double s1, double s2, double s3) : side1(s1), side2(s2), side3(s3) {}

    double area() const override {

        double s = (side1 + side2 + side3) / 2;

        return sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

    }

    double perimeter() const override {

        return side1 + side2 + side3;

    }

    void displayProperties() const override {

        cout << "Properties of the Triangle:" << endl;

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

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

    }

};

class EquilateralTriangle : public Triangle {

public:

    EquilateralTriangle(double side) : Triangle(side, side, side) {}

    void displayProperties() const override {

        cout << "Properties of the Equilateral Triangle:" << endl;

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

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

    }

};

int main() {

    string choice;

    do {

        cout << "Please select a shape:" << endl;

        cout << "1. Circle" << endl;

        cout << "2. Rectangle" << endl;

        cout << "3. Square" << endl;

        cout << "4. Triangle" << endl;

        cout << "Enter your choice: ";

        int shapeChoice;

        cin >> shapeChoice;

        Shape\* shapePtr = nullptr;

        switch (shapeChoice) {

            case 1: {

                double radius;

                cout << "Enter the radius of the circle: ";

                cin >> radius;

                shapePtr = new Circle(radius);

                break;

            }

            case 2: {

                double length, width;

                cout << "Enter the length of the rectangle: ";

                cin >> length;

                cout << "Enter the width of the rectangle: ";

                cin >> width;

                shapePtr = new Rectangle(length, width);

                break;

            }

            case 3: {

                double side;

                cout << "Enter the side length of the square: ";

                cin >> side;

                shapePtr = new Square(side);

                break;

            }

            case 4: {

                double side1, side2, side3;

                cout << "Enter the lengths of the three sides of the triangle: ";

                cin >> side1 >> side2 >> side3;

                shapePtr = new Triangle(side1, side2, side3);

                break;

            }

            default:

                cout << "Invalid choice!" << endl;

        }

        if (shapePtr != nullptr) {

            shapePtr->displayProperties();

            delete shapePtr;

        }

        cout << "Do you want to calculate properties for another shape? (yes/no): ";

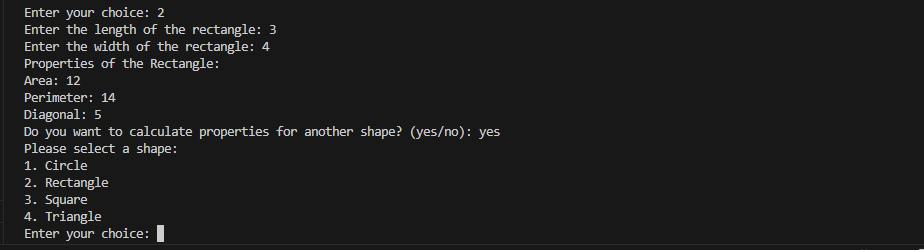
        cin >> choice;

    } while (choice == "yes");

    cout << "Thank you for using the Geometry Competition Calculator!" << endl;

    return 0;

}



Question# 3:

#include <iostream>

#include <string>

using namespace std;

class Employee {

protected:

    int employeeID;

    string employeeName;

public:

    Employee(int id,string name) : employeeID(id), employeeName(name) {}

    virtual ~Employee() {}

    virtual double calculatePay() const {

        return 0.0;

    }

    virtual void displayDetails() const {

        cout << "Employee ID: " << employeeID << "Employee Name: " << employeeName << endl;

    }

};

class FullTimeEmployee : public Employee {

private:

    double monthlySalary;

public:

    FullTimeEmployee(int id,string name, double salary)

        : Employee(id, name), monthlySalary(salary) {}

    double calculatePay() const override {

        return monthlySalary;

    }

    void displayDetails() const override {

        cout << "Employee ID: " << employeeID << "Employee Name: " << employeeName

             << "Monthly Salary: " << monthlySalary << endl;

    }

};

class PartTimeEmployee : public Employee {

private:

    double hourlyWage;

    double hoursWorked;

public:

    PartTimeEmployee(int id,string name, double wage, double hours)

        : Employee(id, name), hourlyWage(wage), hoursWorked(hours) {}

    double calculatePay() const override {

        return hourlyWage \* hoursWorked;

    }

    void displayDetails() const override {

        cout << "Employee ID: " << employeeID << ", Name: " << employeeName

             << "Hourly Wage:" << hourlyWage

             << " Hours Worked: " << hoursWorked << endl;

    }

};

int main() {

    FullTimeEmployee fullTimeEmp(101, "AAA", 5000.0);

    PartTimeEmployee partTimeEmp(102, "BBB", 15.0, 40.0);

    cout << "Displaying Details:" << endl;

    fullTimeEmp.displayDetails();

    partTimeEmp.displayDetails();

    cout << "Full-time Employee Pay:" << fullTimeEmp.calculatePay() << endl;

    cout << "Part-time Employee Pay:" << partTimeEmp.calculatePay() << endl;

    cout << "Using base class pointer to calculate Full-time Employee Pay: $";

    Employee\* empPtr = &fullTimeEmp;

    cout << empPtr->calculatePay() << endl;

    return 0;

}

