#include<iostream>

using namespace std;

//

//class Person {

//public:

// virtual void whoAmI() {

// cout << "I am a person" << endl;

// }

//

// virtual string getClassName() {

// return "Person";

// }

//};

//

//class Superman :public Person {

// void whoAmI()override {

// cout << "I am a superman" << endl;

// }

// string getClassName() override {

// return "Superman";

// }

//};

//

//class Batman :public Person {

// void whoAmI()override {

// cout << "I am a batman" << endl;

// }

// string getClassName() override {

// return "Batman";

// }

//};

//

//

//

//void main() {

// auto persons = new Person \* [3]{

// new Superman,

// new Person,

// new Batman

// };

// for (size\_t i = 0; i < 3; i++)

// {

// persons[i]->whoAmI();

// }

//

//}

#pragma region Abstraction

class Figure {

protected:

double area=0;

public:

virtual void whatIsMyShape() = 0;//pure virtual method

virtual void calculateArea() = 0;//pure virtual method

double GetArea()const {

return area;

}

};

class Rectangle :public Figure {

double length;

double width;

public:

Rectangle(double length,double width)

:length(length),width(width){}

virtual void whatIsMyShape() override

{

cout << "My shape is rectangle" << endl;

}

virtual void calculateArea() override

{

area = length \* width;

}

};

class Square :public Rectangle {

double side;

public:

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

void whatIsMyShape()override {

cout << "My shape is square" << endl;

}

};

class Circle :public Figure {

double radius;

public:

Circle(double radius):radius(radius){}

// Inherited via Figure

virtual void whatIsMyShape() override

{

cout << "My shape is circular" << endl;

}

virtual void calculateArea() override

{

area = 3.141516 \* radius \* radius;

}

};

void main() {

//Figure\* figure = new Figure();//error because figure is abstract class

Figure\* rectangle = new Rectangle(10, 20);

Figure\* circle = new Circle(10);

Figure\* square = new Square(20);

Figure\*\* figures = new Figure \* [3]{ rectangle,circle,square };

for (size\_t i = 0; i < 3; i++)

{

figures[i]->whatIsMyShape();

figures[i]->calculateArea();

cout << "Area : " << figures[i]->GetArea() << endl;

}

//Figure\* f;

/\*Rectangle r(10,20);

r.whatIsMyShape();

r.calculateArea();

cout << r.GetArea() << endl;\*/

}

#pragma endregion