**OOP2**

Mã SV: 19IT182

Tên: Phạm Dương Minh Nhật

# **Class Point**

public abstract class Point

{

protected double position\_x;

protected double position\_y;

public Point(double a, double b)

{

position\_x = a;

position\_y = b;

}

public Point() { }

public double Position\_x

{

get

{

return position\_x;

}

set

{

position\_x = value;

}

}

public double Position\_y

{

get

{

return position\_y;

}

set

{

position\_y = value;

}

}

public void Move(double a, double b)

{

Position\_x += a;

Position\_y += b;

}

public abstract void Show();

}

# **Class Line**

public class Line : Point

{

// private Point endpoint;

private double x\_endpoint;

private double y\_endpoint;

// private Point second;

public double X\_endpoint

{

get

{

return x\_endpoint;

}

set

{

x\_endpoint = value;

}

}

public double Y\_endpoint

{

get

{

return y\_endpoint;

}

set

{

y\_endpoint = value;

}

}

public Line() { }

public Line(double xfirst, double yfirst, double xsecond, double ysecond)

: base(xfirst, yfirst)

{

x\_endpoint = xsecond;

Y\_endpoint = ysecond;

}

public static Line Input()

{

Console.WriteLine("Enter Your x0:");

double xfirst = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Your y0:");

double yfirst = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Your x1:");

double xsecond = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Your y1:");

double ysecond = Convert.ToDouble(Console.ReadLine());

return new Line(xfirst, yfirst, xsecond, ysecond);

}

public override void Show()

{

Console.WriteLine("Your Line");

// base.Show();

Console.WriteLine($"O({Position\_x},{Position\_y})");

Console.WriteLine($"A({X\_endpoint},{Y\_endpoint})");

}

}

# **Class Circle**

public class Circle : Point

{

private double radius;

public double Radius

{

get

{

return radius;

}

set

{

radius = value;

}

}

public Circle() { }

public Circle(double x, double y, double r) : base(x, y)

{

Radius = r;

}

public static Circle Input()

{

Console.WriteLine("Enter Your Ox:");

double x = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Your Oy:");

double y = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Your Radius:");

double r = Convert.ToDouble(Console.ReadLine());

return new Circle(x, y, r);

}

public override void Show()

{

Console.WriteLine("Your Circle");

Console.WriteLine($"O({Position\_x},{Position\_y})");

Console.WriteLine($"Radius: {Radius}");

}

}

# **Class Regtangle**

public class Regtangle : Point

{

private double x\_second;

private double y\_second;

private double x\_third;

private double y\_third;

public double X\_second

{

get

{

return x\_second;

}

set

{

x\_second = value;

}

}

public double Y\_second

{

get

{

return y\_second;

}

set

{

y\_second = value;

}

}

public double X\_third

{

get

{

return x\_third;

}

set

{

x\_third = value;

}

}

public double Y\_third

{

get

{

return y\_third;

}

set

{

y\_third = value;

}

}

public Regtangle() { }

public Regtangle(double xfirst, double yfirst, double xsecond, double ysecond, double xthird, double ythird)

: base(xfirst, yfirst)

{

X\_second = xsecond;

Y\_second = ysecond;

X\_third = xthird;

Y\_third = ythird;

}

public static Regtangle Input()

{

Console.WriteLine("Enter Coordinate for the first point: ");

Console.WriteLine("Enter x0:");

double xfirst = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter y0:");

double yfirst = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter x1:");

double xsecond = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter y1:");

double ysecond = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("enter x3:");

double xthird = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter y3:");

double ythird = Convert.ToDouble(Console.ReadLine());

return new Regtangle(xfirst, yfirst, xsecond, ysecond, xthird, ythird);

}

public override void Show()

{

Console.WriteLine("Regtangle");

// base.Show();

Console.WriteLine($"A({Position\_x},{Position\_y})");

Console.WriteLine($"B({X\_second},{Y\_second})");

Console.WriteLine($"C({X\_third},{Y\_third})");

}

}

# **Class Polyline**

public class PolyLine : Point

{

private List<Double> x;

private List<Double> y;

public List<Double> X

{

get

{

return x;

}

}

public List<Double> Y

{

get

{

return y;

}

}

public void addx(double value)

{

X.Add(value);

}

public void addy(double value)

{

Y.Add(value);

}

public PolyLine() { }

public PolyLine(double x0, double y0, List<Double> px, List<Double> py) : base(x0, y0)

{

x = px;

y = py;

}

public override void Show()

{

Console.WriteLine("PolyLine");

Console.WriteLine($"O({Position\_x},{Position\_y})");

for (var i = 0; i < X.Count; i++)

{

Console.WriteLine($"{(char)('A' + i)}({X[i]},{Y[i]})");

}

}

public static PolyLine Input()

{

int i = 0;

List<Double> x = new List<double>() { };

List<Double> y = new List<double>() { };

Console.WriteLine($"Enter x{i}:");

double x0 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine($"Enter y{i}:");

double y0 = Convert.ToDouble(Console.ReadLine());

i++;

Console.WriteLine($"Enter x{i}:");

x.Add(Convert.ToDouble(Console.ReadLine()));

Console.WriteLine($"Enter y{i}:");

y.Add(Convert.ToDouble(Console.ReadLine()));

int choice;

while (true)

{

i++;

Console.WriteLine("Press 1 to enter the next point");

Console.WriteLine("Press 2 to exit entering");

choice = Convert.ToInt32(Console.ReadLine());

if (choice == 2)

break;

switch (choice)

{

case 2:

break;

case 1:

Console.WriteLine($"Enter x{i}:");

x.Add(Convert.ToDouble(Console.ReadLine()));

Console.WriteLine($"Enter y{i}:");

y.Add(Convert.ToDouble(Console.ReadLine()));

break;

default:

break;

}

}

return new PolyLine(x0, y0, x, y);

}

}

# **Class Animal**

public abstract class Animal

{

private string type;

public string Type

{

get

{

return type;

}

set

{

type = value;

}

}

public Animal() { }

public Animal(string t)

{

Type = t;

}

public abstract void Sound();

public virtual void Information()

{

Console.WriteLine($"Type: {Type}");

}

}

# **Class Dog**

public class Dog : Animal

{

private List<string> breedTypes = new List<string>()

{ "Fox", "Chihuahua", "Shiba" };

private string name;

private int breed;

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public List<string> BreedTypes

{

get

{

return breedTypes;

}

}

public int Breed

{

get { return breed; }

set

{

if (BreedTypes.Contains(BreedTypes[value]))

{

breed = value;

}

}

}

public Dog() : base("mammal") { }

public Dog(string dname, int breedId) : base("mammal")

{

Name = dname;

breed = breedId;

}

public override void Sound()

{

Console.WriteLine($"{Name} says woof woof woof !~~ ");

}

public override void Information()

{

Console.WriteLine("Dog Information");

Console.WriteLine($"Name: {Name}");

base.Information();

Console.WriteLine($"Breed: {BreedTypes[breed]}");

Sound();

}

public void Input()

{

Console.WriteLine("Enter name:");

Name = Console.ReadLine();

Console.WriteLine("Enter breed for your dog(number):");

for (int i = 0; i < BreedTypes.Count; i++)

{

Console.WriteLine($"{i}.{BreedTypes[i]}");

}

Breed = Convert.ToInt16(Console.ReadLine());

}

}

# **Class Cat**

public class Cat : Animal

{

private string name;

private int climb;

private List<string> climbable = new List<string>()

{ "tree", "wall", "roof" };

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public int Climb

{

get

{

return climb;

}

set

{

if (climbable.Contains(climbable[value]))

{

climb = value;

}

}

}

public List<string> Climbable

{

get

{

return climbable;

}

}

public Cat() : base("mammal") { }

public Cat(string cname, int cclimb) : base("mammal")

{

name = cname;

climb = cclimb;

}

public override void Sound()

{

Console.WriteLine($"{name} says Meow Meow Nah Nah!!~~~ !!");

}

public override void Information()

{

Console.WriteLine("Cat Information");

Console.WriteLine($"Name: {Name}");

base.Information();

Console.WriteLine($"{Name} can climb on the {Climbable[climb]}");

Sound();

}

public void Input()

{

Console.WriteLine("Enter name:");

Name = Console.ReadLine();

Console.WriteLine("Which thing your cat can climb on?:(number):");

for (int i = 0; i < climbable.Count; i++)

{

Console.WriteLine($"{i}.{Climbable[i]}");

}

Climb = Convert.ToInt16(Console.ReadLine());

}

}

# **Class Duck**

public class Duck : Animal

{

private string name;

private int swim;

private List<string> swimable = new List<string>() { "pool", "pond", "lake" };

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public int Swim

{

get

{

return swim;

}

set

{

if (swimable.Contains(swimable[value]))

{

swim = value;

}

}

}

public Duck() : base("bird") { }

public Duck(string dname, int dswim) : base("bird")

{

name = dname;

swim = dswim;

}

public override void Sound()

{

Console.WriteLine($"{Name} says quack quack quack ~~~!");

}

public override void Information()

{

Console.WriteLine("Duck Information");

Console.WriteLine($"Name: {Name}");

base.Information();

Console.WriteLine($"{Name} can swim in the {swimable[swim]}");

Sound();

}

public void Input()

{

Console.WriteLine("Enter name:");

Name = Console.ReadLine();

Console.WriteLine("Which your cat can swim in?:(number):");

for (int i = 0; i < swimable.Count; i++)

{

Console.WriteLine($"{i}.{swimable[i]}");

}

Swim = Convert.ToInt16(Console.ReadLine());

}

}

# **Program**

class Program

{

static void Main(string[] args)

{

while (true)

{

Console.WriteLine("Welcome to Minh Nhat EXERCISES OOP2 ABSTRACT");

Console.WriteLine("1.Create a Line");

Console.WriteLine("2.Create a Circle");

Console.WriteLine("3.Create a Regtangle");

Console.WriteLine("4.Create a PolyLine");

Console.WriteLine("5.Present a Dog");

Console.WriteLine("6.Present a Cat");

Console.WriteLine("7.Present a Duck");

Console.WriteLine("8. Exit");

Console.WriteLine("Enter Your Choice Function:");

int chooseFunc = Convert.ToInt32(Console.ReadLine());

if (chooseFunc > 8 || chooseFunc < 1)

{

Console.WriteLine("Please re-Enter Function");

chooseFunc = Convert.ToInt32(Console.ReadLine());

}

else

{

switch (chooseFunc)

{

case 1:

Line line = Line.Input();

line.Show();

break;

case 2:

Circle circlle = Circle.Input();

circlle.Show();

break;

case 3:

Regtangle regtangle = Regtangle.Input();

regtangle.Show();

break;

case 4:

PolyLine polyline = PolyLine.Input();

polyline.Show();

break;

case 5:

Dog dog = new Dog();

dog.Input();

dog.Information();

break;

case 6:

Cat cat = new Cat();

cat.Input();

cat.Information();

break;

case 7:

Duck duck = new Duck();

duck.Input();

duck.Information();

break;

default:

break;

}

}

if (chooseFunc == 8)

{

break;

}

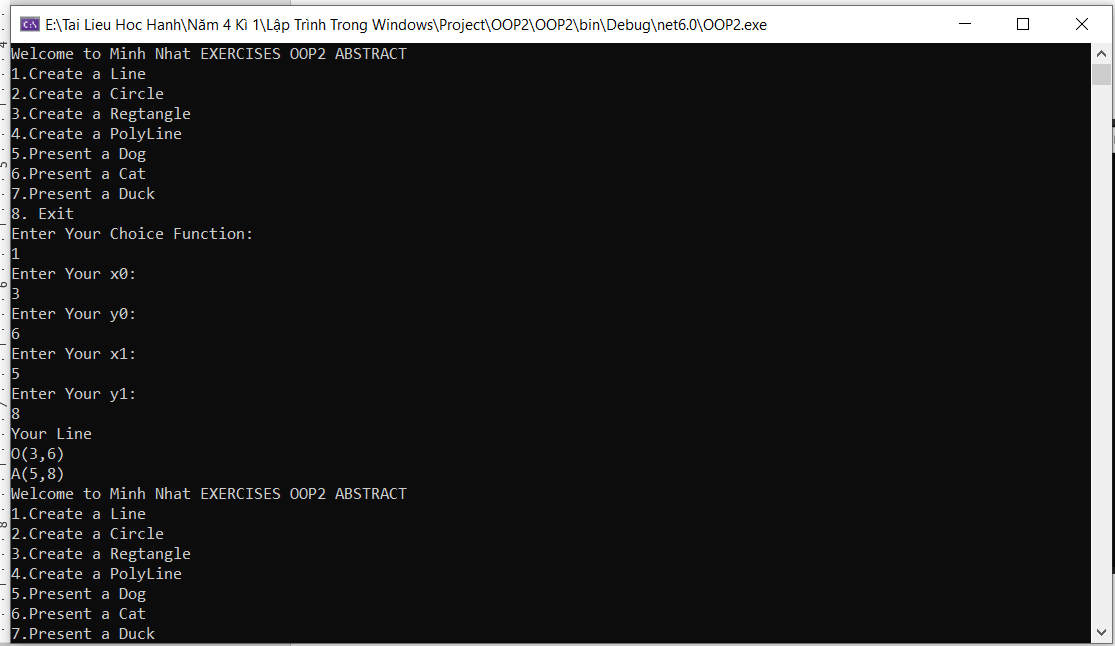
}

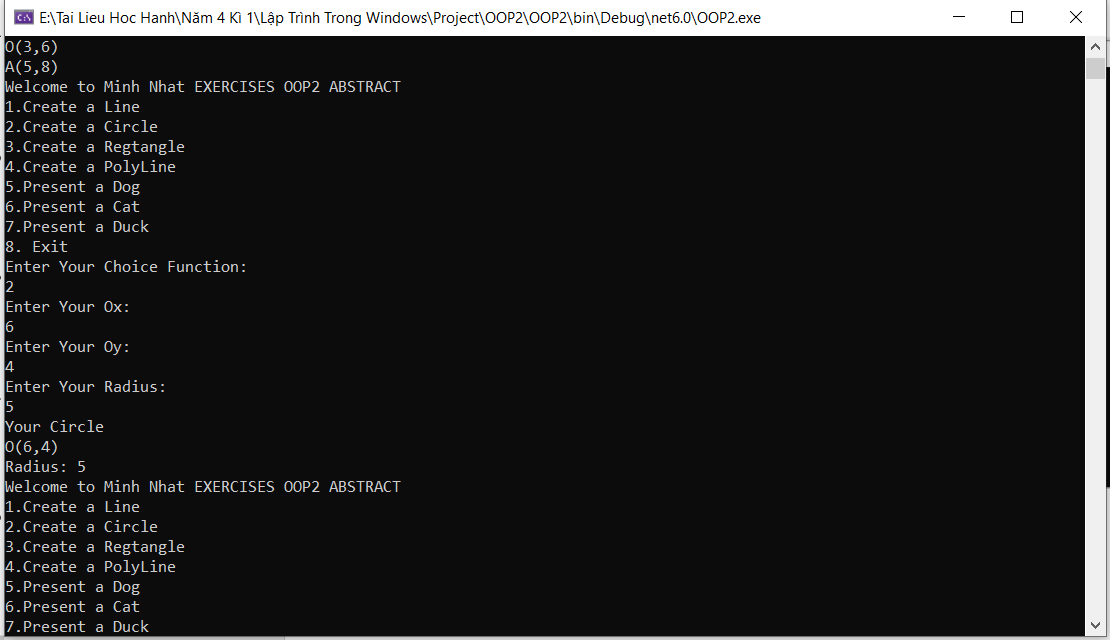
}

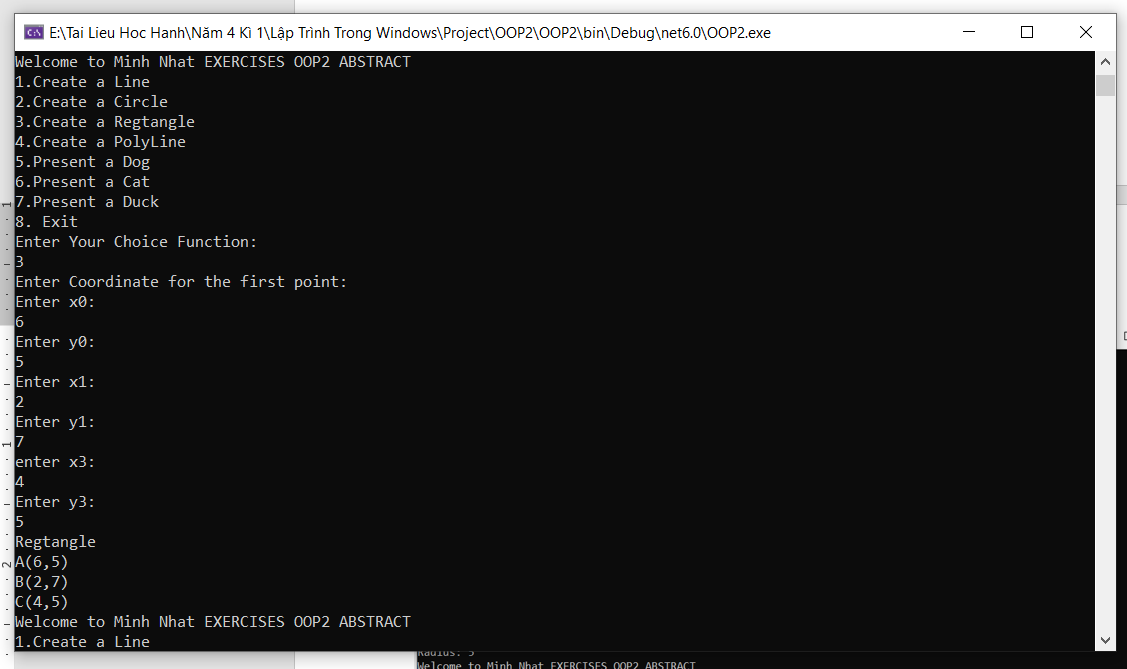
}

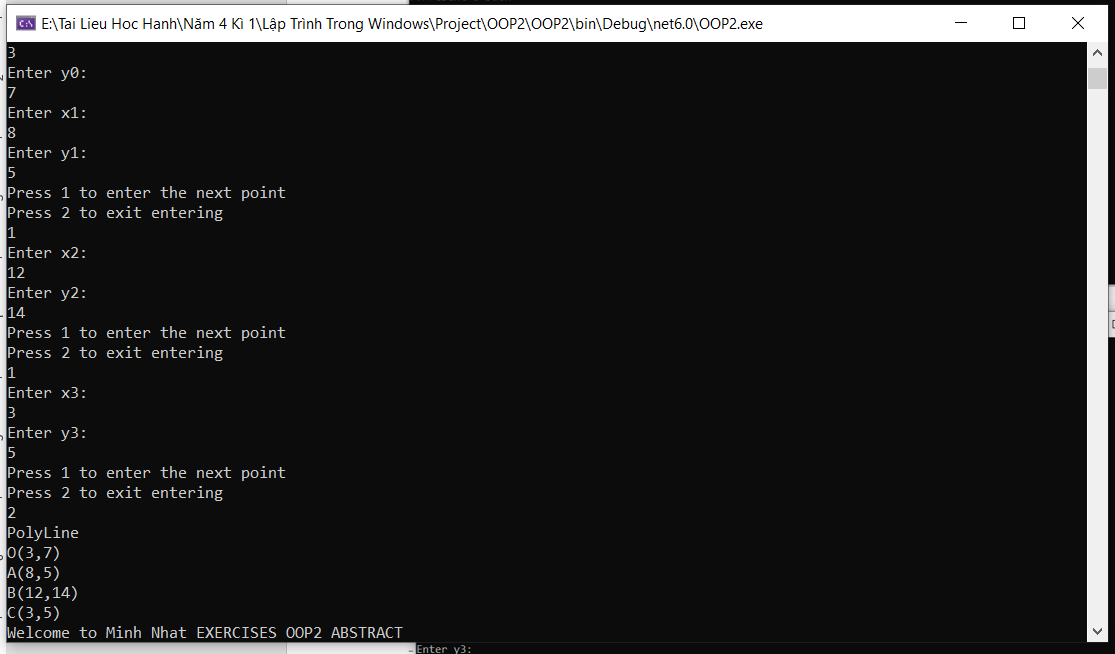
# **Product**

Ex1:









Ex2:

