11. Кондратюк Данило

IDrow interface

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_9\_1

{

interface IDraw

{

void Draw();

}

}

Main task:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab\_9\_1

{

class Program

{

abstract class Triangle :IDraw

{

public string color { get; set; }

public int side1 { get; set; }

public int side2 { get; set; }

public int angle { get; set; }

public string name { get; set; }

public string type { get; set; }

public abstract double Perimeter();

public abstract double Square();

public void Draw()

{

if (color == "red" || color == "Red")

Console.ForegroundColor = ConsoleColor.Red;

if (color == "green" || color == "Green")

Console.ForegroundColor = ConsoleColor.Green;

if (color == "blue" || color == "Blue")

Console.ForegroundColor = ConsoleColor.Blue;

if (color == "white" || color == "White")

Console.ForegroundColor = ConsoleColor.White;

if (color == "magenta" || color == "Magenta")

Console.ForegroundColor = ConsoleColor.Magenta;

if (color == "gray" || color == "Gray")

Console.ForegroundColor = ConsoleColor.Gray;

if (color == "yellow" || color == "Yellow")

Console.ForegroundColor = ConsoleColor.Yellow;

}

public override string ToString()

{

Draw();

return ("Type = "+type+" Side1 = " + side1 + "| Side2 = " + side2 + "| Angle = " + angle + "| Color = " + color + "| Name = " + this.name + "\nPerimeter = " + Perimeter() + "| Square = " + Square());

}

protected void RandomValues()

{

string[] colors = new string[] { "Red", "White", "Yellow", "Green", "Blue", "Magenta","Grey" };

Random random = new Random();

string[] nameArray = new string[] {"d3","H1","u2","ka","s8","q0","b2","iq","ter","p"};

name = nameArray[random.Next(0, nameArray.Length - 1)];

side1 = random.Next(1, 10);

side2 = random.Next(1, 10);

angle = random.Next(1, 180);

color = colors[random.Next(0, colors.Length - 1)];

}

}

class EqTriangle : Triangle

{

public EqTriangle()

{

RandomValues();

side2 = side1;

type = "Equilateral triangle";

Perimeter();

Square();

}

public EqTriangle(string name)

{

RandomValues();

type = "Equilateral triangle";

this.name = name;

Perimeter();

Square();

}

public EqTriangle(string name, int side1)

{

RandomValues();

type = "Equilateral triangle";

this.name = name;

this.side1 = side1;

Perimeter();

Square();

}

public EqTriangle(string name, int side1,int side2)

{

RandomValues();

type = "Equilateral triangle";

this.name = name;

this.side1 = side1;

Perimeter();

Square();

RandomValues();

}

public EqTriangle(string name, int side1, int side2, string color)

{

RandomValues();

type = "Equilateral triangle";

this.name = name;

side2 = side1;

this.side1 = side1;

this.color = color;

Perimeter(); Square();

}

public override double Perimeter()

{

return 3 \* side1;

}

public override double Square()

{

return (0.5 \* side1 \* ((Math.Sqrt(3) / 2) \* side1));

}

}

class IsTriangle : Triangle

{

public IsTriangle()

{

RandomValues();

type = "Isosceles triangle";

Perimeter();

Square();

}

public IsTriangle(string name)

{

RandomValues();

type = "Isosceles triangle";

this.name = name;

Perimeter();

Square();

}

public IsTriangle(string name, int side1)

{

RandomValues();

type = "Isosceles triangle";

this.name = name;

this.side1 = side1;

Perimeter();

Square();

}

public IsTriangle(string name, int side1, int side2)

{

RandomValues();

type = "Isosceles triangle";

this.name = name;

this.side1 = side1;

this.side2 = side2;

Perimeter();

Square();

}

public IsTriangle(string name, int side1, int side2, string color)

{

RandomValues();

type = "Isosceles triangle";

this.name = name;

this.side1 = side1;

this.side2 = side2;

this.color = color;

Perimeter();

Square();

}

public override double Perimeter()

{

return ((side1\*2)+side2);

}

public override double Square()

{

double h = Math.Sqrt(Math.Pow(side1, 2)-(Math.Pow(side2, 2)/4));

return (0.5 \* side2 \* h);

}

}

class RectTriangle : Triangle

{

public RectTriangle()

{

RandomValues();

type = "Rect triangle";

Perimeter();

Square();

}

public RectTriangle(string name)

{

RandomValues();

type = "Rect triangle";

this.name = name;

Perimeter();

Square();

}

public RectTriangle(string name, int side1)

{

RandomValues();

type = "Rect triangle";

this.name = name;

this.side1 = side1;

Perimeter();

Square();

}

public RectTriangle(string name, int side1, int side2)

{

RandomValues();

type = "Rect triangle";

this.name = name;

this.side1 = side1;

this.side2 = side2;

Perimeter();

Square();

}

public RectTriangle(string name, int side1, int side2, string color)

{

RandomValues();

type = "Rect triangle";

this.name = name;

this.side1 = side1;

this.side2 = side2;

this.color = color;

Perimeter();

Square();

}

public override double Perimeter()

{

double length3 = Math.Sqrt(Math.Pow(side1, 2) + Math.Pow(side2, 2));

return (side1 + side2 + length3);

}

public override double Square()

{

return ((side1\*side2)/2);

}

}

class Picture

{

List<Triangle> picture = new List<Triangle>();

private void GetChoice()

{

ConsoleKeyInfo userAction;

Console.WriteLine("\n\nIf you went to add a triangle press \t'A'\nIf you went to delete a triangle pres \t'D'");

userAction = Console.ReadKey(true);

Console.Clear();

switch (userAction.Key)

{

case ConsoleKey.A:

Console.WriteLine("Enter parameters of triangle in form like this: name,side1,side2,color\n\nSide1 and side2 its must be integer.\nIf you do not want to use one of these parameters, just do not add to it, but follow the sequence! ");

string allParametrs = Console.ReadLine();

char type;

WhichType(out type);

AddTraingle(type,allParametrs);

break;

case ConsoleKey.D:

GetPicture();

DeleteTraingle();

break;

}

}

private void WhichType(out char type)

{

ConsoleKeyInfo userAction;

Console.WriteLine("If you want to add a rect triangle press \t'R'\nIf you want to add a isosceles triangle press \t'I'\nIf you want to add a equilat triangle press \t'E'");

userAction = Console.ReadKey(true);

Console.Clear();

switch (userAction.Key)

{

default:

case ConsoleKey.R:

type = 'R';

break;

case ConsoleKey.I:

type = 'I';

break;

case ConsoleKey.E:

type = 'E';

break;

}

}

public Picture()

{

picture.Add(new EqTriangle() { name = "E1", side1 = 6, side2 = 6 });

picture.Add(new IsTriangle() { side1 = 31, side2 = 2, color = "Red" });

picture.Add(new IsTriangle() { side1 = 11, side2 = 22, color = "Blue" });

picture.Add(new RectTriangle() { name = "Re3", side1 = 19 });

picture.Add(new RectTriangle() { name = "Peq", side1 = 10 });

GetPicture();

GetChoice();

GetPicture();

}

public Picture(int length)

{

int i = 0;

char type;

while (i < length)

{

WhichType(out type);

if (type == 'R')

picture.Add(new RectTriangle());

else if(type == 'I')

picture.Add(new IsTriangle());

else if(type == 'E')

picture.Add(new EqTriangle());

i ++;

}

GetPicture();

GetChoice();

GetPicture();

}

public void GetPicture()

{

Console.WriteLine();

int i = 0;

foreach (Triangle uc in picture)

{

Console.WriteLine("Index: " + (i++)+"\t");

Console.WriteLine(uc);

Console.ResetColor();

}

}

public void AddTraingle(char type,string allParameters)

{

string[] arrayAllParametrs = allParameters.Split(new char[] { ',' });

int size = arrayAllParametrs.Length;

switch (type)

{

case 'R':

switch (size)

{

case 1:

picture.Add(new RectTriangle(arrayAllParametrs[0]));

break;

case 2:

picture.Add(new RectTriangle(arrayAllParametrs[0], Convert.ToInt32(arrayAllParametrs[1])));

break;

case 3:

picture.Add(new RectTriangle(arrayAllParametrs[0], Convert.ToInt32(arrayAllParametrs[1]), Convert.ToInt32(arrayAllParametrs[2])));

break;

case 4:

default:

picture.Add(new RectTriangle(arrayAllParametrs[0], Convert.ToInt32(arrayAllParametrs[1]), Convert.ToInt32(arrayAllParametrs[2]), arrayAllParametrs[3]));

break;

}

break;

case 'I':

switch (size)

{

case 1:

picture.Add(new IsTriangle(arrayAllParametrs[0]));

break;

case 2:

picture.Add(new IsTriangle(arrayAllParametrs[0], Convert.ToInt32(arrayAllParametrs[1])));

break;

case 3:

picture.Add(new IsTriangle(arrayAllParametrs[0], Convert.ToInt32(arrayAllParametrs[1]), Convert.ToInt32(arrayAllParametrs[2])));

break;

case 4:

default:

picture.Add(new IsTriangle(arrayAllParametrs[0], Convert.ToInt32(arrayAllParametrs[1]), Convert.ToInt32(arrayAllParametrs[2]), arrayAllParametrs[3]));

break;

}

break;

case 'E':

switch (size)

{

case 1:

picture.Add(new EqTriangle(arrayAllParametrs[0]));

break;

case 2:

picture.Add(new EqTriangle(arrayAllParametrs[0], Convert.ToInt32(arrayAllParametrs[1])));

break;

case 3:

picture.Add(new EqTriangle(arrayAllParametrs[0], Convert.ToInt32(arrayAllParametrs[1]), Convert.ToInt32(arrayAllParametrs[2])));

break;

case 4:

default:

picture.Add(new EqTriangle(arrayAllParametrs[0], Convert.ToInt32(arrayAllParametrs[1]), Convert.ToInt32(arrayAllParametrs[2]), arrayAllParametrs[3]));

break;

}

break;

}

Console.ReadLine();

GetChoice();

}

public void DeleteTraingle()

{

ConsoleKeyInfo userAction;

Console.WriteLine("\n\nTo delete a triangle by name press \t'N'by type \t'T' by square\t'S' ");

userAction = Console.ReadKey(true);

switch (userAction.Key)

{

case ConsoleKey.N:

Console.WriteLine("Enter name of triangle ");

string parameter = Console.ReadLine();

int j = 0;

while (j < picture.Count)

{

if (picture[j].name == parameter)

{

picture.RemoveAt(j);

}

else { j++; }

}

Console.Clear();

GetPicture();

Console.WriteLine("\n\nTo delete a triangle by name press \t'N'by type \t'T' by square\t'S' \nEnter name of triangle \n"+parameter);

break;

case ConsoleKey.T:

char type;

WhichType(out type);

switch (type)

{

case 'R':

default:

picture.Remove(new RectTriangle());

break;

case 'E':

picture.Remove(new EqTriangle());

break;

case 'I':

picture.Remove(new IsTriangle());

break;

}

break;

case ConsoleKey.S:

Console.WriteLine("\n\nEnter a square. triangles with a square larger than yours will be removed ");

parameter = Console.ReadLine();

j = 0;

while (j < picture.Count)

{

if (picture[j].Square() > Convert.ToInt32(parameter))

{

picture.RemoveAt(j);

}

else { j++; }

}

Console.Clear();

GetPicture();

Console.WriteLine("\n\nEnter a square. triangles with a square larger than yours will be removed " + parameter);

break;

}

GetChoice();

}

}

class Painter

{

}

static void Main(string[] args)

{

ConsoleKeyInfo userAction;

Console.WriteLine("if you want to add a certain number of triangles press \t'C'");

userAction = Console.ReadKey(true);

if (userAction.Key == ConsoleKey.C)

{

Console.WriteLine("Enter рow many triangles do you want to add?");

new Picture(Convert.ToInt32(Console.ReadLine()));

}

else

{

new Picture();

}

Console.ReadKey();

}

}

}