Assignment 4

Source code

*MainApp.cs*

using lab4;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Console;

namespace lab4

{

internal class MainApp

{

static void Main(string[] args)

{

int switch\_input;

Write("Chap 4: Programming Exercises: 1, 3 " +

"\nChap 5: Programming Exercises: 5, 6" +

"\nChap 6: Programming Exercises: 7, 9" +

"\nChoose assignment: ");

if (int.TryParse(ReadLine(), out switch\_input))

{

switch (switch\_input)

{

case 1:

{

Date();

break;

}

case 3:

{

Motorway();

break;

}

case 5:

{

Shipping();

break;

}

case 6:

{

Area();

break;

}

case 7:

{

Multiplication\_table();

break;

}

case 9:

{

Triangle();

break;

}

default:

{

WriteLine("Invalid input");

break;

}

}

}

else { WriteLine("Invalid input"); }

ReadKey();

}

static void Date()

{

Write("Input day: ");

int day = Convert.ToInt32(ReadLine());

Write("Input month: ");

int month = Convert.ToInt32(ReadLine());

Write("Input year: ");

int year = Convert.ToInt32(ReadLine());

Date newDate = new Date(month, day, year);

Write("\nDate you input is ");

WriteLine(newDate);

Write("Input month: ");

string s\_month = ReadLine();

Date newDate1 = new Date(s\_month, day, year);

Write("\nAnother date is ");

Write(newDate1);

}

static void Motorway() {

Motorway fMotorway = new Motorway();

fMotorway.NameOfTheHighway = "Jackson";

fMotorway.TypeOfStreet = "Ave";

fMotorway.Direction = "N";

fMotorway.Surface = "Gravel";

fMotorway.NumberOfLanes = 4;

fMotorway.IsToll = true;

fMotorway.Maintains = "State";

WriteLine("\n\tFirst Motoway\n");

WriteLine(fMotorway);

Motorway sMotorway = new Motorway("Jackson", true, "State");

sMotorway.TypeOfStreet = "Street";

sMotorway.Direction = "E";

sMotorway.Surface = "Asphalt";

sMotorway.NumberOfLanes = 2;

WriteLine("\n\tSecond Motoway\n");

WriteLine(sMotorway);

Motorway tMotorway = new Motorway("Jackson","Blvd","E","Blacktop", 6,false,"Country");

WriteLine("\n\tThird Motoway\n");

WriteLine(tMotorway);

}

public static void Shipping()

{

double standard\_price = 2.99, disc\_price1 = 1.99, disc\_price2 = 1.49, lowest\_price = 0.99;

Write("Enter the # of items: ");

int items = Convert.ToInt32(ReadLine());

if (items == 1)

{

WriteLine("Your shipping charges are: ${0:.##}", standard\_price);

}

else if ((items >= 2) && (items <= 5))

{

WriteLine("Your shipping charges are: ${0:.##}", (standard\_price + (items - 1) \* disc\_price1));

}

else if ((items > 5) && (items < 15))

{

WriteLine("Your shipping charges are: ${0:.##}", (standard\_price) + (4 \* disc\_price1) + ((items - 5) \* disc\_price2));

}

else

{

WriteLine("Your shipping charges are: ${0:.##}", (standard\_price + (4 \* disc\_price1) + ((10) \* disc\_price2) + ((items - 14) \* lowest\_price)));

}

}

public static void Area()

{

double radius, length, width, height;

WriteLine("Please select a figure" +

"\n1)Circle" +

"\n2)Rectangle" +

"\n3)Cylinder");

int input = Convert.ToInt32(ReadLine());

if (input == 1) {

Write("Please input radius: ");

radius = Convert.ToDouble(ReadLine());

Figure figure = new Circle(radius);

WriteLine(string.Format(string.Format("Area = {0:.##}", figure.Area())));

}

else if (input == 2) {

Write("Please input length: ");

length = Convert.ToDouble(ReadLine());

Write("Please input width: ");

width = Convert.ToDouble(ReadLine());

Figure figure = new Rectangle(length, width);

WriteLine(string.Format(string.Format("Area = {0:.##}", figure.Area())));

}

else {

Write("Please input radius: ");

radius = Convert.ToDouble(ReadLine());

Write("Please input height: ");

height = Convert.ToDouble(ReadLine());

Figure figure = new Cylinder(radius, height);

WriteLine(string.Format(string.Format("Area = {0:.##}", figure.Area())));

}

}

private static void Multiplication\_table()

{

string inValue, inValue2;

int first\_base, last\_base;

Write("Enter the first base value: ");

inValue = ReadLine();

Write("Enter the second base value: ");

inValue2 = ReadLine();

if (int.TryParse(inValue, out first\_base) && int.TryParse(inValue2, out last\_base)

&& (first\_base < last\_base) && first\_base >= 2 && last\_base <= 8)

{

for (int i = 1; i <= 25; i++)

{

Write(i + "\t");

for (int j = first\_base; j <= last\_base; j++)

{

if (i > 0) { Write(i \* j + "\t"); }

else { Write(j + "\t"); }

}

Write("\n");

}

}

else {

WriteLine("Invalid input");

}

}

private static void Triangle()

{

int size;

Write("Enter the size of the triangle: ");

if (!int.TryParse(ReadLine(), out size) || size > 10)

{size = 3;}

Write("Enter the character of the triangle: ");

string character = ReadLine();

if (character.Length > 1 || Char.IsDigit(character[0]))

{character = "\*";}

for (int i = 1; i <= size; i++)

{

for (int j = 0; j < i; j++)

{

Write(character);

}

WriteLine();

}

for (int i = size - 1; i >= 1; i--)

{

for (int j = 0; j < i; j++)

{

Write(character);

}

WriteLine();

}

}

}

}

*Date.cs*

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab4

{

internal class Date

{

private int year;

public int Year {

get { return year; }

set { year = value; }

}

private int month;

public int Month {

get { return month; }

set { month = value; }

}

private int day;

public int Day {

get { return day; }

set { day = value; }

}

private string month\_string;

public string Month\_string {

get { return month\_string; }

set { month\_string = value; }

}

public Date() { }

public Date(int \_month, int \_day, int \_year)

{

this.year = \_year;

this.month = \_month;

this.day = \_day;

this.month\_string = getNameOfMonth();

}

public Date(string \_month, int \_day, int \_year)

{

this.year = \_year;

this.month\_string = \_month;

this.day = \_day;

this.month = DateTime.ParseExact(\_month, "MMMM", CultureInfo.CurrentCulture).Month;

}

public override string ToString()

{

return month + "/" + day + "/" + year + " or " + getNameOfMonth() + " " + day + ", " + year;

}

public string getNameOfMonth()

{

string mName = new DateTime(year, month, day).ToString("MMMM");

return mName;

}

}

}

*Figure.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab4

{

internal abstract class Figure

{

public abstract double Area();

}

class Circle : Figure

{

private double radius;

public double Radius

{

get { return radius; }

set { radius = value; }

}

public Circle(double \_radius) {

this.radius = \_radius;

}

public override double Area()

{

return Math.PI \* radius \* radius;

}

}

class Rectangle : Figure

{

private double length;

public double Length

{

get { return length; }

set { length = value; }

}

private double width;

public double Width

{

get { return width; }

set { width = value; }

}

public Rectangle(double \_length, double \_width)

{

this.length = \_length;

this.width = \_width;

}

public override double Area()

{

return length \* width;

}

}

class Cylinder : Figure

{

private double radius;

public double Radius

{

get { return radius; }

set { radius = value; }

}

private double height;

public double Height

{

get { return height; }

set { height = value; }

}

public Cylinder(double \_radius, double \_height)

{

this.radius = \_radius;

this.height = \_height;

}

public override double Area()

{

return (2 \* Math.PI \* radius \* height + 2 \* Math.PI \* Math.Pow(radius,2));

}

}

}

*Motorway.cs*

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace lab4

{

internal class Motorway

{

private string nameOfTheHighway;

public string NameOfTheHighway {

get { return nameOfTheHighway; }

set { nameOfTheHighway = value; }

}

private string typeOfStreet;

public string TypeOfStreet {

get { return typeOfStreet; }

set { typeOfStreet = value; }

}

private string direction;

public string Direction {

get { return direction; }

set { direction = value; }

}

private string surface;

public string Surface {

get { return surface; }

set { surface = value; }

}

private int numberOfLanes;

public int NumberOfLanes {

get { return numberOfLanes; }

set { numberOfLanes = value; }

}

private bool isToll;

public bool IsToll {

get { return isToll; }

set { isToll = value; }

}

private string maintains;

public string Maintains {

get { return maintains; }

set { maintains = value; }

}

public Motorway() { }

public Motorway(string \_name, bool \_toll, string \_maintains)

{

this.nameOfTheHighway = \_name;

this.isToll = \_toll;

this.maintains = \_maintains;

}

public Motorway(string \_name,string \_type,string \_dir, string \_sur, int \_lanes, bool \_toll, string \_maintains) {

this.nameOfTheHighway = \_name;

this.typeOfStreet = \_type;

this.direction = \_dir;

this.surface = \_sur;

this.numberOfLanes = \_lanes;

this.isToll = \_toll;

this.maintains = \_maintains;

}

public override string ToString()

{

return "Name of Motoway: " + getFullNameOfMotorway()

+ "\nSurface: " + surface

+ "\nNumber of Lanes: " + numberOfLanes

+ "\nToll: " + isToll

+ "\nMaintained by: " + maintains;

}

public string getFullNameOfMotorway() {

return nameOfTheHighway + " " + typeOfStreet + " " + direction;

}

public string getFullNameOfMotorwayAndToll()

{

return "Name of Motoway: " + getFullNameOfMotorway()

+ "\nisToll: " + isToll;

}

public string getFullNameOfMotorwayAndNumberOfLanes()

{

return "Name of Motoway: " + getFullNameOfMotorway()

+ "\nNumber of Lanes: " + numberOfLanes;

}

}

}

Outputs

4.1)

Text

Description automatically generated

4.3)

Text

Description automatically generated

5.5)

Text

Description automatically generated

5.6)

Text

Description automatically generated

6.7)

A picture containing calendar

Description automatically generated

6.9)

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated