通識計算機程式設計期末考

6/24/2016

試題共 7 題，兩面印製 28 頁，滿分 103

1. 參考下頁 UML 類別圖(圖 1)，撰寫 C#敘述達成下列要求: (假設 using System; 敘述已經包含於程式中)。可想像這是未來物聯網(internet of things, IOT)中，以智慧型手機畫面上的按鈕(buttons)遙控家中空調及燈光的設施的概念雛形。本題只要求完成圖 1 類別圖的一部分。

   (a) 撰寫介面 **SwitchableDevice**，其中包含函式宣告 **TurnOn()**、**TurnOff()**、 **Status()**、**Name()**。其中的列舉型別 **SwitchStatus** 定義為: **enum SwitchStatus { ON, OFF }** 。 (3%)

   (b) 由圖 1 可知類別程式 **AirConditioner** 實作 **SwitchableDevice**。撰寫其建構式 **AirConditioner(string name)**：設定其名稱、**status** 設為 **SwitchStatus.OFF**，**temperatureSetting** 為 28，並建立 **rand** (不用外來種子數)。 (6%)

   (c) 在類別 **AirConditioner** 中實作 **public 的成員函式 TurnOn()**，虛擬碼為：

```
public void TurnOn() {
    if(偵測到的溫度低於溫度設定)
        return;
    if(status == SwitchStatus.OFF) {
        切換 status;
    }
}
```

   (6%)

   (d) 撰寫類別 **Lamp** 建構式 **Lamp(string name)** 及其成員函式 **TurnOn()**、**TurnOff()**、**Status()**、**Name()**。(3%)

   (e) 實作類別 **Button** 建構式 **Button(SwitchableDevice device)**，設定其 **status** 為 **ButtonStatus.RELEASED** 及所控制的 **SwitchableDevice**。列舉型別 **ButtonStatus** 定義為：**enum ButtonStatus { PRESSED, RELEASED }** 。 (6%)

   (f) 寫一段測試主控台主程式，建立兩個元素的 **Button** 陣列 **buttons**，令 **buttons[0]**控制名稱為"**AC_LivingRoom**"的 **AirConditioner** 物件，**buttons[1]**控制名為"**Lamp_Porch**"的 **Lamp** 物件。隨後依序呼叫 **buttons[0]**及 **buttons[1]**的 **Pressed()**函式。**Button** 類別中的

Pressed() 函式內容如下：

```
public void Pressed() {
    device.TurnOn();
    status = ButtonStatus.PRESSED;
    Console.WriteLine("{0} is turned on", device.Name());
}
```

此段主控台主程式之執行畫面如圖 2。(6%)

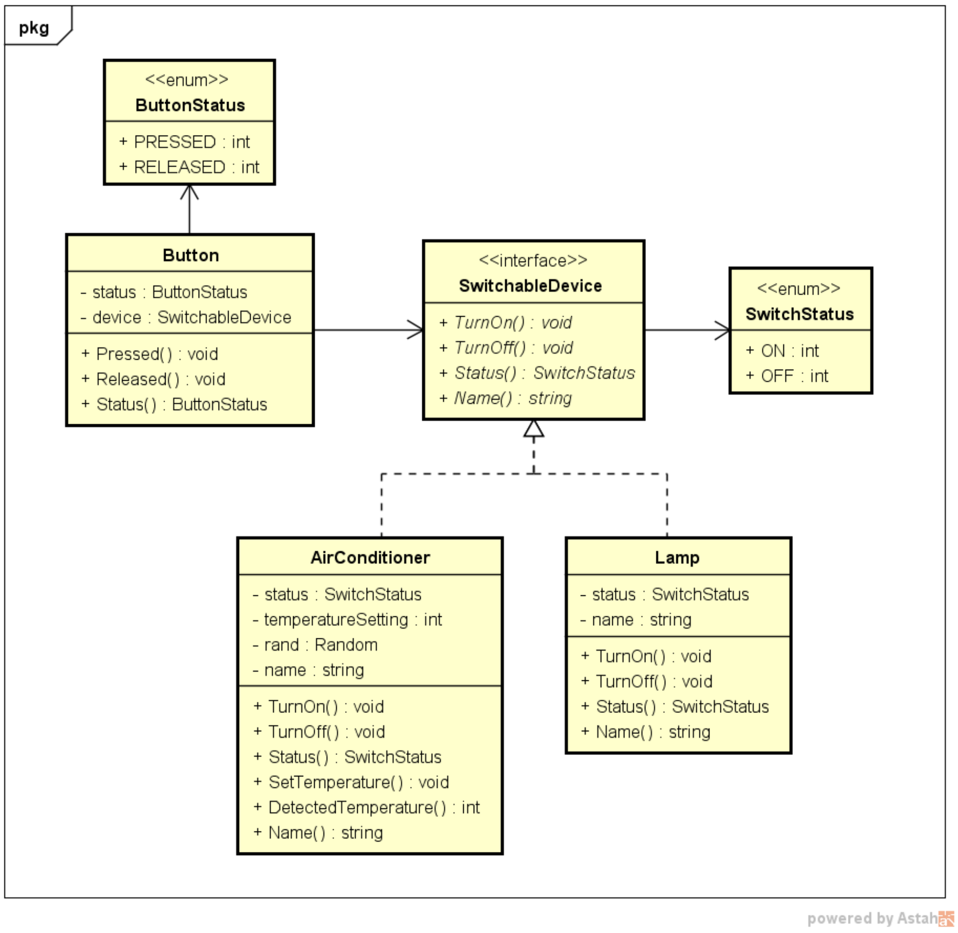
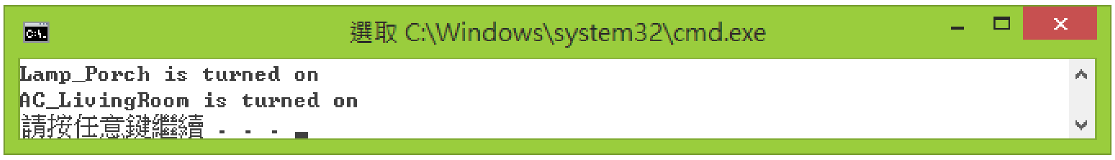


圖 1. 第 1 題對應的 UML 類別圖



```
Lamp_Porch is turned on
AC_LivingRoom is turned on
請按任意鍵繼續 . . .
```

圖 2. 第 1(f)題程式碼執行主控台螢幕畫面

2. 找出以下程式片段之錯誤，並予更正.

(a) (3%) 一個錯誤

```
class Rectangle
```

```
{
    private int width;
    private int height;
    public Rectangle(int w, int h)
    {
        width = w;
        height = h;
    }
    public int Width
    {
        get { return width; }
    }
    public int Height
    {
        get { return height; }
    }
}
class Program
{
    static void Main(string[] args)
    {
        Rectangle rect = Rectangle(4, 6);
    }
}
```

(b) (3%) 一個錯誤

```
struct ComplexNumber
{
    public double real;
    public double imag;
    public ComplexNumber()
    {
        real = 0.0;
        imag = 0.0;
    }
    public ComplexNumber(double re, double im)
    {
        real = re;
        imag = im;
    }
}
```

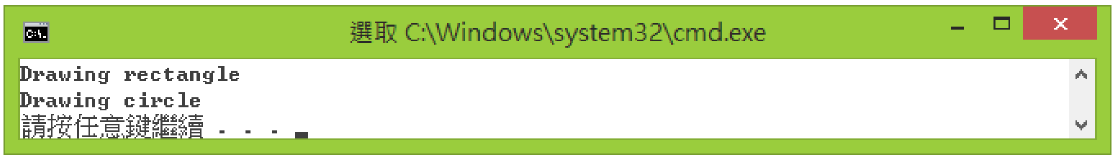
(c) (3%) 一種錯誤。

```
class Plane
{
    void Fly()
    {
      Console.WriteLine("Fly");
    }
}
class Ship
{
    void Sail()
    {
      Console.WriteLine("Sail");
    }
}
class Floatplane : Plane, Ship
{
     public Floatplane()
    {
      Console.WriteLine("建立水上飛機");
    }
}
```

(d) (3%) 一種錯誤

```
class Shape
{
    private string type;
    public Shape(string type)
    {
        this.type = type;
    }
    virtual public void Draw()
    {
        Console.WriteLine("Drawing shape");
    }
}

class Rectangle : Shape
{
    private int width;
    private int height;
    public Rectangle(int w, int h) : base("rectangle")
    {
        width = w;
        height = h;
    }
    public void Draw()
    {
        Console.WriteLine("Drawing rectangle");
    }
}
class Circle : Shape
{
    private int radius;
    public Circle(int r) : base("circle")
    {
        radius = r;
    }
    public void Draw()
    {
        Console.WriteLine("Drawing circle");
    }
}
class Program
{
    static void Main(string[] args)
    {
        Shape[] shapes = new Shape[2];
        shapes[0] = new Rectangle(8, 6);
        shapes[1] = new Circle(5);

        // 應該達成”多型”的效果
```

```
            for(int i=0; i<2; ++i)
            {
                shapes[i].Draw();
            }
        }
```



選取 C:\Windows\system32\cmd.exe

```
Drawing rectangle
Drawing circle
請按任意鍵繼續 . . .
```

圖3. 第2(d)題改正後，應有的主控台程式執行畫面

(e) (3%) 一種錯誤

```
class Rectangle
{
    private int width;
    private int height;
    private int nRectangles = 0;
    public Rectangle()
    {
        width = 1;
        height = 1;
        ++nRectangles;
    }
    public Rectangle(int w, int h)
    {
        width = w;
        height = h;
        ++nRectangles;
    }
    public int Width
    {
        get { return width; }
    }
    public int Height
    {
        get { return height; }
    }
    public int NRectangles()
```

```
        {
            return nRectangles;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Rectangle rec1 = new Rectangle();
            Rectangle rec2 = new Rectangle(4, 6);
            // 應該顯示一共產生了多少個矩形
            Console.WriteLine(Rectangle.NRectangles());
        }
    }
```

3. 試寫出下列程式的輸出 (12%)

```
// Problem3.Program
using System;
using System.Collections.Generic;
namespace Problem3
{
    class Program
    {
        static void Main(string[] args)
        {
            Graph konigsberg = new Graph();
            konigsberg.AddNode(1);
            konigsberg.AddNode(2);
            konigsberg.AddNode(3);
            konigsberg.AddNode(4);
            konigsberg.AddEdge(1, 2);
            konigsberg.AddEdge(1, 2);
            konigsberg.AddEdge(1, 3);
            konigsberg.AddEdge(1, 4);
            konigsberg.AddEdge(1, 4);
            konigsberg.AddEdge(2, 3);
            konigsberg.AddEdge(3, 4);
            List<GraphNode> oddNodes;
            List<GraphNode> evenNodes;
            konigsberg.ClassifyNodes(out oddNodes, out evenNodes);
            Console.Write("Odd nodes: { ");
            OutputNodes(oddNodes);
            Console.WriteLine("}");
            Console.Write("Even nodes: { ");
            OutputNodes(evenNodes);
            Console.WriteLine("}");
        }
        static void OutputNodes(List<GraphNode> nodes)
        {
            foreach (GraphNode node in nodes)
            {
                Console.Write(node.Value + " ");
            }
        }
    }
}

// Problem3.Graph
using System;
using System.Collections.Generic;
namespace Problem3
{
    public class Graph
    {
        private List<GraphNode> nodeSet;
        public Graph()
        {
            nodeSet = new List<GraphNode>();
        }
        public void AddNode(int value)
        {
            nodeSet.Add(new GraphNode(value));
        }
        public void AddEdge(GraphNode from, GraphNode to)
        {
            from.AddNeighbor(to);
            to.AddNeighbor(from);
        }
        public void AddEdge(int fromLabel, int toLabel)
        {
            GraphNode from = null;
            foreach(GraphNode node in nodeSet)
            {
                if (node.Value == fromLabel)
                {
                    from = node;
                    break;
                }
            }
            GraphNode to = null;
            foreach(GraphNode node in nodeSet)
            {
                if(node.Value == toLabel)
                {
                    to = node;
                    break;
                }
            }
            if (from == null || to == null) return;
            AddEdge(from, to);
        }
        public List<GraphNode> Nodes
        {
            get { return nodeSet; }
        }
        public int Count
        {
           get { return nodeSet.Count; }
        }
        public void ClassifyNodes(out List<GraphNode> oddNodes,
              out List<GraphNode> evenNodes)
        {
            evenNodes = new List<GraphNode>();
            oddNodes = new List<GraphNode>();
```

```
            foreach(GraphNode node in nodeSet)
            {
                if (node.Degree() % 2 == 0)
                {
                    evenNodes.Add(node);
                }
                else
                {
                    oddNodes.Add(node);
                }
            }
        }
    }
}

// Problem3.GraphNode
using System;
using System.Collections.Generic;
namespace Problem3
{
    public class GraphNode
    {
        int value;
        List<GraphNode> neighbors;
        public GraphNode(int value)
        {
            this.value = value;
            neighbors = new List<GraphNode>(); ;
        }
        public void AddNeighbor(GraphNode neighbor)
        {
            neighbors.Add(neighbor);
        }
        public int Value
        {
            get { return value; }
        }
        public List<GraphNode> Neighbors
        {
            get { return neighbors; }
        }
        public int Degree()
        {
            return neighbors.Count;
        }
    }
}
```

4. 試寫出以下程式在下列狀況時的主控台螢幕輸出。

(a) (3%) 檔案 **pyramid.stl** 尚未建立。

(b) (3%) 檔案 **pyramid.stl** 已在正確位置，且內容為

```
solid pyramid
facet normal 0.0 0.0 -1.0
outer loop
vertex 0.0 0.0 0.0
vertex 0.0 1.0 0.0
vertex 1.0 1.0 0.0
endloop
endfacet
outer loop
vertex 0.0 0.0 0.0
vertex 1.0 1.0 0.0
vertex 1.0 0.0 0.0
endloop
endfacet
endsolid
```

(c) (3%) 檔案 **pyramid.stl** 已在正確位置，且內容為

```
solid pyramid
facet normal 0.0 0.0 -1.0
outer loop
vertex 0.0 0.0 0.0
vertex 0.0 1.0 0.0
vertex 1.0 1.0 0.0
endloop
endfacet
```

```
facet normal 0.0 0.0 -1.0
outer loop
vertex 0.0 0.0 0.0
vertex 1.0 1.0 0.0
vertex 1.0 0.0 0.0
endloop
endfacet
```

(d) (3%) 檔案 **`pyramid.stl`** 已在正確位置，且內容為

```
    solid pyramid
    facet normal 0.0 0.0 -1.0
    outer loop
    vertex 0.0 0.0 0.0
    vertex 0.0 1.0 0.0
    vertex 1.0 1.0 0.0
    endloop
    endfacet
    facet normal 0.0 0.0 -1.0
    outer loop
    vertex 0.0 0.0 0.0
    vertex 1.0 1.0 0.0
    vertex 1.0 0.0 0.0
    endloop
    endfacet
    facet normal 0.0 0.707107 0.707107
    outer loop
    vertex 1.0 1.0 0.0
    vertex 0.0 1.0 0.0
    vertex 0.5 0.5 0.5
    endloop
    endfacet
    facet normal -0.707107 0.0 0.70107
    outer loop
    vertex 0.0 1.0 0.0
    vertex 0.0 0.0 0.0
    vertex 0.5 0.5 0.5
    endloop
    endfacet
    facet normal 0.0 -0.707107 0.70107
    outer loop
    vertex 0.0 0.0 0.0
    vertex 1.0 0.0 0.0
    vertex 0.5 0.5 0.5
    endloop
    endfacet
    facet normal 0.707107 0.0 0.70107
    outer loop
    vertex 1.0 0.0 0.0
    vertex 1.0 1.0 0.0
    vertex 0.5 0.5 0.5
    endloop
    endfacet
    endsolid
```

===

```
// Problem4.Program
using System;

namespace Problem4
{
   class Program
   {
      static void Main(string[] args)
      {
         try
         {
            Console.WriteLine("輸入STL檔名, 包含.stl副檔名");
            string fileName = Console.ReadLine();
            Solid_STL_ASCII_Parser parser = new
               Solid_STL_ASCII_Parser(fileName);

            Solid solid = parser.STL_Solid;
            Console.WriteLine("solid " + solid.Name);
            solid.WriteFacets();
         }
         catch(Exception e)
```

```
            {
                Console.WriteLine("主程式接到一個exception");
            }
        }
    }
}

// Problem4.Solid_STL_ASCII_Parser
using System;
using System.Collections.Generic;
using System.IO;
namespace Problem4
{
    public class Solid_STL_ASCII_Parser
    {
        private Solid stl_solid;
        public Solid_STL_ASCII_Parser(string fileName)
        {
            try
            {
                StreamReader reader = new StreamReader(fileName);
                Console.WriteLine(fileName + " 開啟");
                try
                {
                    string header = reader.ReadLine();
                    string[] heads = new string[2];
                    heads = header.Split(' ');
                    if (heads[0] != "solid")
                    {
                        Console.WriteLine(
                           "Invalid STL header, expected \"solid [name]\"");
                        throw new FormatException(
                           "Invalid STL header, expected \"solid [name]\"");
                    }
                    string solidName = heads[1];

                    Facet_STL_ASCII_Parser facetParser = new
                        Facet_STL_ASCII_Parser(reader);
                    List<Facet> facets = new List<Facet>();
                    Facet facet = null;
                    while (!reader.EndOfStream)
                    {
                        facet = facetParser.Read();
                        if(facet != null)
                        {
                            facets.Add(facet);
                        }
                    }
                    stl_solid = new Solid(solidName, facets);
                }
                catch(FormatException e)
                {
                    Console.WriteLine(
                       "由Solid_STL_ASCII_Parser內圈拋出FormatException");
                    throw e;
                }
                catch (IOException e)
                {
                    Console.WriteLine(
                        "由Solid_STL_ASCII_Parser內圈拋出IOException");
                    throw e;
                }
                catch (Exception e)
                {
                    Console.WriteLine(
                        "由Solid_STL_ASCII_Parser內圈拋出Exception");
                    throw e;
                }
                finally
                {
                    reader.Close();
                    Console.WriteLine(fileName + " 關閉");
                }
            }
            catch (FileNotFoundException e)
            {
```

```
                Console.WriteLine(
                "由Solid_STL_ASCII_Parser外圈拋出FileNotFoundException");
                throw e;
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "由Solid_STL_ASCII_Parser外圈拋出Exception");
                throw e;
            }
        }
        public Solid STL_Solid
        {
            get { return stl_solid; }
        }
    }
}

// Problem4.Facet_STL_ASCII_Parser
using System;
using System.Collections.Generic;
using System.IO;
namespace Problem4
{
    public class Facet_STL_ASCII_Parser
    {
        private StreamReader reader;
        public Facet_STL_ASCII_Parser(StreamReader reader)
        {
            this.reader = reader;
        }
        public Facet Read()
        {
            try
            {
                string line = reader.ReadLine();
                if (line == "endsolid" || line == "end solid") return null;
                string[] terms = new string[5];
        terms = line.Split(' ');
        if (terms[0] != "facet" || terms[1] != "normal")
        {
            Console.WriteLine(
"Invalid STL facet data, expected \"facet normal [normal vector]\"");
            throw new FormatException(
"Invalid STL facet data, expected \"facet normal [normal vector]\"");
        }
        Vector3D normal;
        normal.x = double.Parse(terms[2]);
        normal.y = double.Parse(terms[3]);
        normal.z = double.Parse(terms[4]);

        // skip outer loop
        reader.ReadLine();

        // read three vertices
        Vector3D[] vertices = new Vector3D[3];
        string[] vTerms = new string[4];
        for (int i = 0; i < 3; ++i)
        {
            line = reader.ReadLine();
            vTerms = line.Split(' ');
            if (vTerms[0] != "vertex")
            {
                Console.WriteLine(
    "Invalid STL facet data, expected \"vertex [vertex vector]\"");
                throw new FormatException(
    "Invalid STL facet data, expected \"vertex [vertex vector]\"");
            }
            vertices[i].x = double.Parse(vTerms[1]);
            vertices[i].y = double.Parse(vTerms[2]);
            vertices[i].z = double.Parse(vTerms[3]);
        }

        // skip two lines:
        // endloop
        // endfacet
```

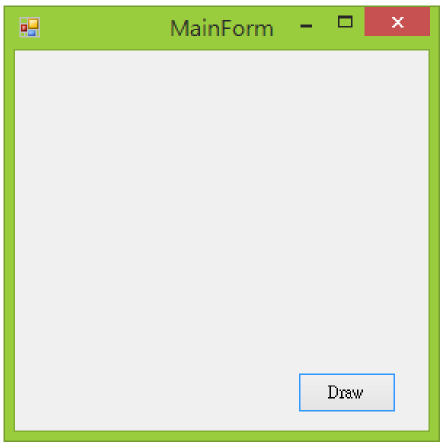
```
                reader.ReadLine();
                reader.ReadLine();

                Facet facet = new Facet(
                  vertices[0], vertices[1], vertices[2], normal);
                return facet;
            }
            catch(FormatException e)
            {
                Console.WriteLine(
                    "由Facet_STL_ASCII_Parser抛出FormatException");
                throw e;
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "由Facet_STL_ASCII_Parser抛出Exception");
                throw e;
            }
        }
    }
}

// Problem4.Solid
using System;
using System.Collections.Generic;
namespace Problem4
{
    public class Solid
    {
        string name;
        private List<Facet> facets;
        public Solid(string name, List<Facet> facets)
        {
            this.name = name;
            this.facets = new List<Facet>();
            CopyFrom(facets);
        }
```

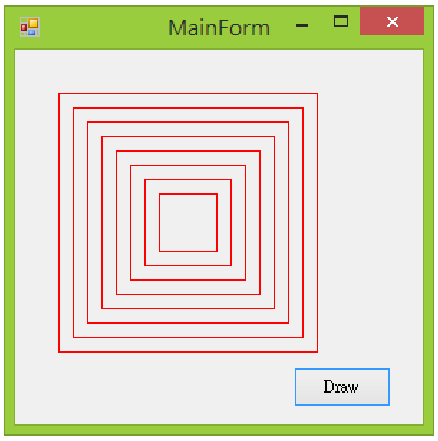
```
public string Name
{
    get { return name; }
}
public List<Facet> Facets()
{
    List<Facet> facetsCopy;
    CopyTo(out facetsCopy);
    return facetsCopy;
}

public void WriteFacets()
{
    int n = 0;
    foreach(Facet facet in facets)
    {
        Console.WriteLine("Facet " + n);
        Console.Write("normal = ");
        Console.WriteLine("[{0}, {1}, {2}]",
            facet.Normal.x, facet.Normal.y, facet.Normal.z);
        Console.Write("vertex_1 = ");
        Console.WriteLine("[{0}, {1}, {2}]",
           facet.Vertex_1.x, facet.Vertex_1.y, facet.Vertex_1.z);
        Console.Write("vertex_2 = ");
        Console.WriteLine("[{0}, {1}, {2}]",
           facet.Vertex_2.x, facet.Vertex_2.y, facet.Vertex_2.z);
        Console.Write("vertex_3 = ");
        Console.WriteLine("[{0}, {1}, {2}]",
           facet.Vertex_3.x, facet.Vertex_3.y, facet.Vertex_3.z);
        ++n;
    }
}

private void CopyFrom(List<Facet> facets)
{
    this.facets = new List<Facet>();
    foreach (Facet facet in facets)
```

```
            {
                this.facets.Add(facet);
            }
        }
        private void CopyTo(out List<Facet> facetsCopy)
        {
            facetsCopy = new List<Facet>();
            foreach (Facet facet in facets)
            {
                facetsCopy.Add(facet);
            }
        }
    }
}

// Problem4.Facet
using System;
namespace Problem4
{
    public class Facet
    {
        private Vector3D r1;
        private Vector3D r2;
        private Vector3D r3;
        private Vector3D normal;

        public Facet(Vector3D vertex1, Vector3D vertex2,
                     Vector3D vertex3, Vector3D normal)
        {
            r1 = vertex1;
            r2 = vertex2;
            r3 = vertex3;
            this.normal = normal;
        }
        public Vector3D Vertex_1
        {
            get { return r1; }
        }
        public Vector3D Vertex_2
        {
            get { return r2; }
        }
        public Vector3D Vertex_3
        {
            get { return r3; }
        }
        public Vector3D Normal
        {
            get { return normal; }
        }
    }
}

// Problem4.Vector3D
using System;
namespace Problem4
{
    public struct Vector3D
    {
        public double x;
        public double y;
        public double z;
    }
}
```

5.依據以下描述及程式框架，完成指定程式。你在答案卷只需寫下程式註解標示的部分。(6%)

程式描述 :畫出同心正方形
建立如圖 4 之圖形使用介面，按下「**Draw**」按鈕(**button1**)，即於主視窗出現 8 個同心正方形，如圖 5 所示。最外圈正方形左上角座標為(20, 20)，邊長為 200。每一層正方形的左上角座標比外一層正方形座標，*x* 與 *y* 方向都增加 10，而邊長比外一層正方形邊長少 20。





(左)圖 4. 程式開始執行時之視窗畫面
(右)圖 5. 按下「Draw」按鈕後某時間截取之視窗畫面

```
// Problem5.MainForm
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Problem5
{
   public partial class MainForm : Form
   {
      //****************************
      private bool started = false;
      Point topLeft0;
      int length0;
      Square square;
      //****************************

      public MainForm()
      {
         InitializeComponent();
         //************************
```

```
            topLeft0.x = 20;
            topLeft0.y = 20;
            length0 = 200;
            //************************
        }
        private void button1_Click(object sender, EventArgs e)
        {
            //******************************************
            started = true;
            DrawContents();
            //******************************************
        }
        //********************************************
        private void DrawContents()
        {
            Graphics g = this.CreateGraphics();
            DrawAllSquares(8, 10, g);
            g.Dispose();
        }
        private void DrawAllSquares(int n, int increment, Graphics g)
        {
            // 請依照如下說明完成此處應有的程式敘述 (6%)
            // 畫出n個同心正方形
            // 輸入參數: n 代表要畫的同心正方形數
            //           increment 代表每一層正方形左上角座標比外一層正方形
            //                     左上角座標，在x與y方向都增加的pixel數
            //           g 是視窗程式提供的繪圖工具
        }
        protected override void OnPaint(PaintEventArgs e)
        {
            base.OnPaint(e);
            if (started) DrawContents();
        }
        //********************************************
    }
}
```

```
// Problem5.Square
using System;
using System.Drawing;
namespace Problem5
{
    struct Point
    {
        public int x;
        public int y;
    }
    class Square
    {
        private int length;
        private Point topLeft;
        public Square(int length, Point topLeft)
        {
            this.topLeft = topLeft;
            this.length = length;
        }
        public void Draw(Graphics g)
        {
            g.DrawRectangle(new Pen(Color.Red),
                topLeft.x, topLeft.y, length, length);
        }
    }
}
```

5. 3D 列印是當今極受重視的資訊技術，有人還認為它是德國所倡工業 4.0 的基礎之一。3D 列印技術的優勢之一是可以在網路上取得各種部件的 3D 圖檔，以自己的 3D 列印機一層一層地長出其外形。目前最簡單、最普及的 3D 圖檔是 STL (STereoLithography)檔(請參閱 https://en.wikipedia.org/wiki/STL_(file_format))。STL 的格式一般分為 ASCII 文字檔及二進位(binary)檔兩種，ASCII 文字檔較易處理，而二進位檔在部件內容複雜龐大時較有用處。本題採用 STL 的 ASCII 格式，檔案結構如下(進一步假設每行資料都由最左方開始):

solid [部件名稱]
facet normal n_x n_y n_z

```
outer loop
vertex v1x v1y v1z
vertex v2x v2y v2z
vertex v3x v3y v3z
endloop
endfacet
endsolid
```

其中 outerloop 和 endloop 之間是是一個三角形的三個頂點空間座標，而 facet 和 endfacet 之間代表某一個法向量(向著部件外面，垂直於表面的單位向量)對應的平面三角片，endsolid 是部件資料的結束。請注意如此的定義，其實沒有排除 outerloop 與 endloop 之間有超過三個頂點，代表一個平面多邊形的情況，也沒有去除一個 facet 包含有多個三角形或多邊形(即有多個 outerloop 與 endloop 配對，其中有三角形或多邊形頂點)的可能性。不過，本題加以簡化，假定每一個 facet 都是三角片，而一個部件由多個三角片構成，例如：以下命名為 **`pyramid.stl`** 的檔案，其外形如圖 6 (以一個功能較簡單的免費軟體 STL View http://www.freestlview.com 呈現，網址 https://all3dp.com/best-free-stl-file-viewer-online-mac-pc-linux-download-android-ios-app/ 有多達 14 種的免費 STL viewer 介紹)：

```
solid pyramid
facet normal 0.0 0.0 -1.0
outer loop
vertex 0.0 0.0 0.0
vertex 0.0 1.0 0.0
vertex 1.0 1.0 0.0
endloop
endfacet
facet normal 0.0 0.0 -1.0
outer loop
vertex 0.0 0.0 0.0
vertex 1.0 1.0 0.0
vertex 1.0 0.0 0.0
endloop
endfacet
facet normal 0.0 0.707107 0.707107
outer loop
vertex 1.0 1.0 0.0
vertex 0.0 1.0 0.0
vertex 0.5 0.5 0.5
endloop
endfacet
facet normal -0.707107 0.0 0.70107
outer loop
vertex 0.0 1.0 0.0
vertex 0.0 0.0 0.0
vertex 0.5 0.5 0.5
endloop
endfacet
facet normal 0.0 -0.707107 0.70107
outer loop
vertex 0.0 0.0 0.0
vertex 1.0 0.0 0.0
vertex 0.5 0.5 0.5
endloop
endfacet
facet normal 0.707107 0.0 0.70107
outer loop
vertex 1.0 0.0 0.0
vertex 1.0 1.0 0.0
vertex 0.5 0.5 0.5
endloop
endfacet
endsolid
```

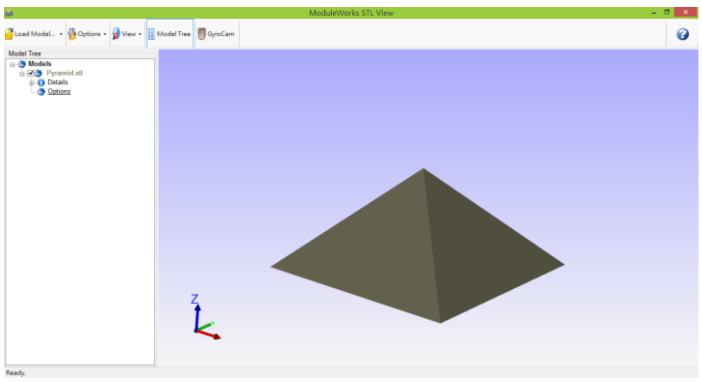


圖 6. STL 檔 **pyramid.stl** 的三度空間圖形

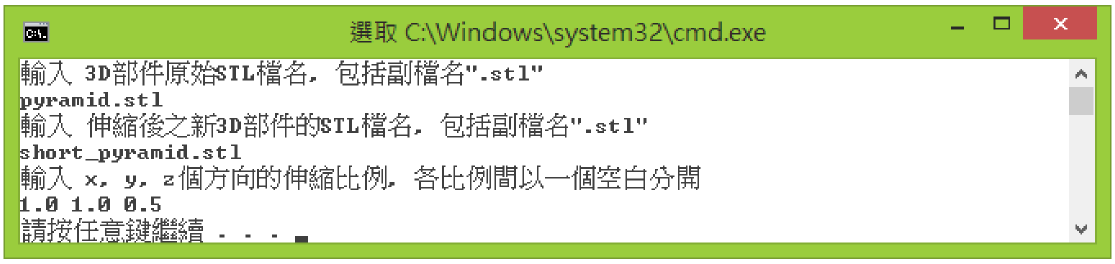
由於網路上找到的 STL 檔，不見得完全符合我們需求，因此我們必須倚賴 STL editor：讀入 STL 檔後，將其修改，再將修改後外形以 STL 格式輸出為另一個 STL

檔。功能最多最複雜的 STL editor，就是一種一般用於機械的計算機輔助設計(Computer-Aided Design, CAD)或用於遊戲及動畫特效製作的 3D 物件編輯器。網址 https://all3dp.com/free-stl-editor-open-edit-stl-file/ 介紹 4 種免費 STL editor 軟體)。

本題要求撰寫一個極度簡化的 STL editor：讀入一個全由三角片構成 facet 的 STL ASCII 檔，將各三角片頂點的 *x*、*y*、*z* 座標分別伸縮 *sx*、*sy*、*sz* 倍(相當於將座標分別乘上 *sx*、*sy*、*sz*)，再寫出對應的 STL 檔。只要處理文字檔案，不必畫圖(交給 STL View 就行)。

程式執行時的主控台畫面一例，如圖 7 所示，對應的輸出檔 short_pyramid.stl 內容如下：

```
solid pyramid
facet normal 0 0 -1
outer loop
vertex 0 0 0
vertex 0 1 0
vertex 1 1 0
endloop
endfacet
facet normal 0 0 -1
outer loop
vertex 0 0 0
vertex 1 1 0
vertex 1 0 0
endloop
endfacet
facet normal 0 0.707107 0.707107
outer loop
vertex 1 1 0
vertex 0 1 0
vertex 0.5 0.5 0.25
endloop
endfacet
facet normal -0.707107 0 0.70107
outer loop
vertex 0 1 0
vertex 0 0 0
vertex 0.5 0.5 0.25
endloop
endfacet
facet normal 0 -0.707107 0.70107
outer loop
vertex 0 0 0
vertex 1 0 0
vertex 0.5 0.5 0.25
endloop
endfacet
facet normal 0.707107 0 0.70107
outer loop
vertex 1 0 0
vertex 1 1 0
vertex 0.5 0.5 0.25
endloop
endfacet
endsolid
```



```
輸入 3D部件原始STL檔名, 包括副檔名".stl"
pyramid.stl
輸入 伸縮後之新3D部件的STL檔名, 包括副檔名".stl"
short_pyramid.stl
輸入 x, y, z個方向的伸縮比例, 各比例間以一個空白分開
1.0 1.0 0.5
請按任意鍵繼續 . . .
```

圖 7. 程式執行之主控台畫面一例

本題滿分 25 分，全部程式集中寫成一個大 **Main** 函式，不區分函式者，最高得 20 分；善用函式者，最高得 23 分；能利用虛擬碼或 UML 類別圖思考，適當劃分類別(class)，運用 Collections 者，最高得 25 分。 (25%)

提示：可以直接使用及修改第 4 題所附程式，重複的部分，只需標記” **略去與第 4 題重複程式碼”** ，不用再抄一遍其內容。

7. 請寫下本課程教學「待改進」之處及改進方法建議。 (3%)