

期末複習五參考解答

試題共 7 題，滿分 100

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)

- (a) 撰寫抽象類別 **Item**，其中包含 **protected** 的 **string** 成員變數 **title**、**protected** 的 **int** 成員變數 **year**、**public** 的抽象成員函式 **void Dump()**。(3%)

答:

```
abstract class Item
{
    protected string title;
    protected int year;
    public abstract void Dump();
}
```

- (b) 撰寫類別程式 **Book**，繼承抽象類別 **Item**，增添 **private** 之 **string** 成員變數 **majorAuthor**。建構式以三個參數輸入 **title**，**year**，及 **majorAuthor** 之值。(6%)

答:

```
class Book : Item
{
    private string majorAuthor;
    public Book(string title, int year, string majorAuthor)
    {
        this.title = title;
        this.year = year;
        this.majorAuthor = majorAuthor;
    }
}
```

- (c) 在類別 **Book** 中實作 **public override void Dump()**，於主控台螢幕顯示三列，每列為一個成員變數之值。(6%)

答:

```
public override void Dump()
{
    Console.WriteLine("Book title:" + title);
    Console.WriteLine("Book publication year: " + year);
    Console.WriteLine("Book major author:" + majorAuthor);
}
```

- (d) 撰寫類別 **Video**，繼承抽象類別 **Item**，增添 **private** 之 **string** 成員變數 **majorDirector**。建構式設定所有成員變數之值。(3%)

答:

```
class Video : Item
{
    private string majorDirector;
    public Video(string title, int year, string majorDirector)
    {
        this.title = title;
        this.year = year;
        this.majorDirector = majorDirector;
    }
}
```

- (e) 實作類別 **Video** 的 **public override void Dump()** 成員函式，於主控台螢幕顯示三列，每列為一個成員變數之值。(6%)

答:

```
public override void Dump()
{
    Console.WriteLine("Video title: " + title);
    Console.WriteLine("Video release year: " + year);
    Console.WriteLine("Video major director: " + majorDirector);
}
```

- (f) 寫一段測試主控台主程式，建立兩個元素的 **Item** 陣列 **items**，令 **items[0]** 為 **Book** 物件，**items[1]** 為 **Video** 物件。**Book** 物件的資料為 "雪國 千鶴 古都"、1994、"川端康成"。**Video** 物件資料為 "變形金剛"、2009、"麥可貝"。隨後依序呼叫 **items[0]** 及 **items[1]** 的 **Dump**

函式，主控台螢幕畫面如圖 2。(6%)

答:

```
static void Main(string[] args)
{
    Item[] items = new Item[2];
    items[0] = new Book("雪國 千鶴 古都", 1994, "川端康成");
    items[1] = new Video("變形金剛", 2009, "麥可貝");
    for(int i = 0; i < items.Length; i++)
    {
        items[i].Dump();
    }
}
```

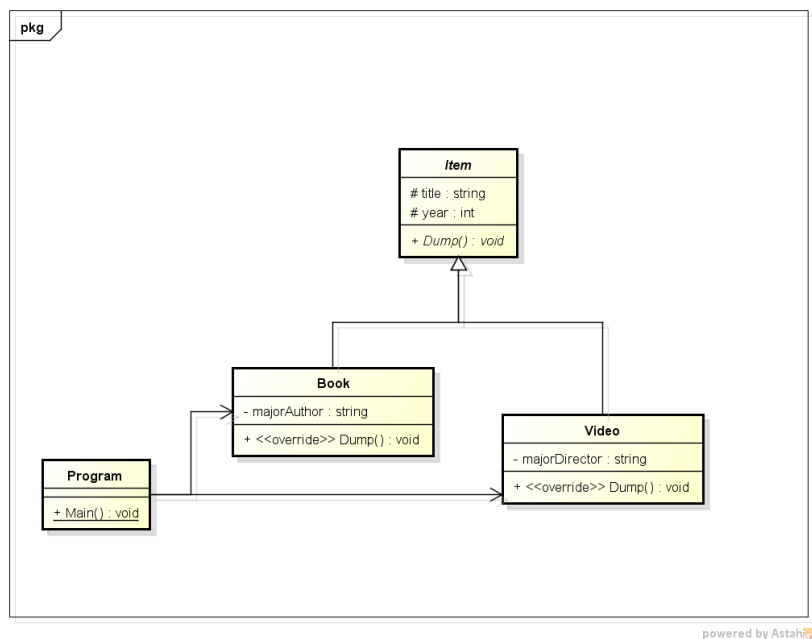


圖 1. 第 1 題對應的 UML 類別圖

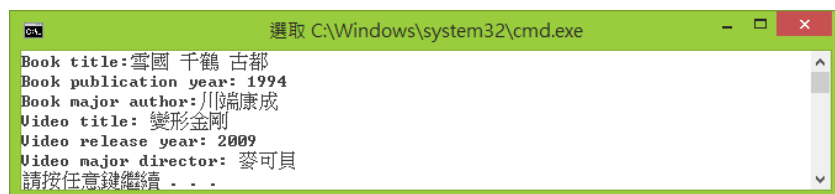


圖 2. 第 1(f)題程式碼執行主控台螢幕畫面

2. 找出以下程式片段之錯誤，並予更正.

(a) (3%) 一種錯誤

答:

Rectangle 的屬性(Property) Width 和 Height 都只能讀取內容而無法設定其值. 改正的方法之一, 是在屬性定義中加入 set 敘述, 即:

```
class Rectangle
{
    private int width;
    private int height;
    public Rectangle()
    {
        width = 0;
        height = 0;
    }
    public int Width
    {
        get { return width; }
        set { width = value; }
    }
    public int Height
    {
        get { return height; }
        set { height = value; }
    }
}
class Program
{
    static void Main(string[] args)
    {
        Rectangle rect = new Rectangle();
        rect.Width = 4;
        rect.Height = 6;
    }
}
```

(b) (3%) 一個錯誤

答:

兩個 **Ratio** 函式名稱, 引數均相同, 僅有回傳變數型別不同(**double** 和 **int**), 這會造成這兩個函式等同為一個函式宣告兩次(分辨函式的 **signature** 只包含名稱及引數數目, 型別, 及位置, 不包括回傳變數的型別), 造成錯誤. 解決方法之一便是更改兩個 **Ratio** 函式之一的名稱, 如:

```
class ComplexNumber
{
    private double real;
    private double imag;
    public ComplexNumber(double re, double im)
    {
        real = re;
        imag = im;
    }
    public double Ratio()
    {
        return imag / real;
    }
    public int Ratio() Quotient()
    {
        return (int)(imag / real);
    }
}
```

(c) (3%) 一個錯誤, 正確的主控台螢幕輸出畫面應該如圖 3 所示。

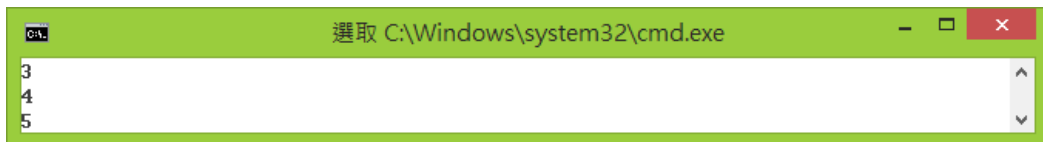


圖 3. 第 2(c)題的主控台螢幕正確畫面

答:

敘述「**Triangle t2 = t1;**」僅完成 shallow copy, 設定 **t2** 和 **t1** 的參考(reference)相同. 因此當 **t1.Rotate** 時, **t1** 內的 **a, b, c** 輪換, 而 **t2** 因為 reference 與 **t1** 相同, 其內的 **a, b, c** 即是 **t1** 輪換後的 **a, b, c**, 因此主控台螢幕輸出不會是原先的 3, 4, 5, 與題目要求不合. 修改的方法便是改用 copy constructor, 即:

```

class Triangle
{
    private int a;
    private int b;
    private int c;
    public Triangle(int a, int b, int c)
    {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    Public Triangle(Triangle t)    // copy constructor
    {
        a = t.a;
        b = t.b;
        c = t.c;
    }

    public void Rotate()
    {
        int temp = a;
        a = b;
        b = c;
        c = temp;
    }

    public int A
    {
        get { return a; }
    }

    public int B
    {
        get { return b; }
    }
}

```

```

    public int C
    {
        get { return c; }
    }
}

class Program
{
    static void Main(string[] args)
    {
        Triangle t1 = new Triangle(3, 4, 5);
Triangle t2 = t1; Triangle t2 = new Triangle(t1);
        t1.Rotate();
        Console.WriteLine(t2.A);
        Console.WriteLine(t2.B);
        Console.WriteLine(t2.C);
    }
}

```

(d) (3%) 一個錯誤

答：

運用多型時，子類別實作父類別的virtual函式，要加上override修飾語，即：

```

class Melody
{
    protected byte[] notes;
    protected int n;
    public Melody(int n)
    {
        this.n = n;
        for(int i = 0; i < n; i++)
        {
            notes[i] = (byte)0;
        }
    }
    public virtual void PutNote(int i)

```

```

    {
    }
}

class HiAndLo : Melody
{
    public HiAndLo(int n) : base(n)
    {
    }

    // 要作為多型之用
    public override void PutNote(int i)
    {
        notes[i] = Convert.ToByte(
            60 + 12*Math.Sin(2.0 * i * Math.PI / n));
    }
}

```

(e) (3%) 一個錯誤

答：

Interface不可以有成員變數，也不能有建構式，只能包含無內容的單純的成員函式宣告。因此本題只需將interface改為class即可。亦即：

```

interface class Clock
{
    private int hour;
    private int minute;
    public Clock(int h, int m)
    {
        hour = h;
        minute = m;
    }
    public void Click()
    {
        if(minute == 59)
        {
            ++hour;

```



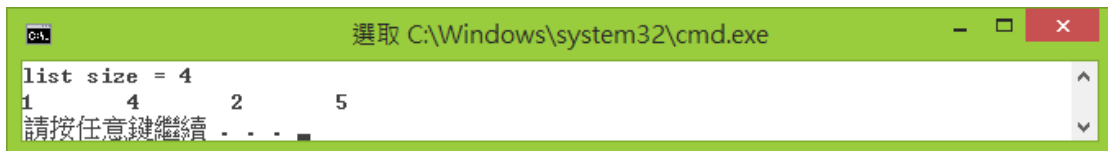
```

        minute = 0;
    }
    else
    {
        minute++;
    }
}
}

```

3. 試寫出下列程式的輸出 (12%)

答:



```

C:\Windows\system32\cmd.exe
list size = 4
1      4      2      5
請按任意鍵繼續 . . .

```

```

// Problem3.Program
using System;

namespace Problem3
{
    class Program
    {
        static void Main(string[] args)
        {
            LinkedList list = new LinkedList();
            list.Add("1");
            list.Add("2");
            list.Add("3");
            list.Insert(1, "4");
            list.Add("5");
            list.Delete(3);
            int size = list.Size;
            Console.WriteLine("list size = " + size);
            for(int n = 0; n < size; n++)

```

```

        {
            Console.Write(list.Data(n) + "\t");
        }
        Console.WriteLine();
    }
}
}

```

```
// Problem3.LinkedList
```

```
using System;
```

```
namespace Problem3
```

```

{
    public class LinkedList
    {
        private Node head = null;
        private int size = 0;

        public LinkedList()
        {
            head = null;
        }

        public LinkedList(string data)
        {
            head = new Node(data, null);
        }

        public int Size
        {
            get { return size; }
        }

        public string Data(int n)
        {
            if (n < 0 || n >= size) return null;
            Node node = head;
            for (int i = 0; i < n; i++)

```

```

        {
            node = node.next;
        }
        return node.data;
    }

    public void Add(string data)
    {
        if(size <= 0)
        {
            head = new Node(data, null);
            size = 1;
        }
        else
        {
            Node tail = Tail();
            Node newTail = new Node(data, null);
            tail.next = newTail;
            size++;
        }
    }

    private Node Tail()
    {
        if(size <= 0)
        {
            return null;
        }
        else
        {
            Node node = head;
            while (node.next != null)
            {
                node = node.next;
            }
            return node;
        }
    }
}

```

```

public void Insert(int n, string data)
{
    if (n < 0 || n >= size) return;
    Node node = head;
    for (int i = 1; i < n; i++)
    {
        node = node.next;
    }
    Node nodeToInsert = new Node(data, node.next);
    node.next = nodeToInsert;
    size++;
}

public void Delete(int n)
{
    if (n < 0 || n >= size) return;
    if(n == 0)
    {
        head = null;
        size = 0;
    }
    else
    {
        Node node = head;
        for (int i = 1; i < n; i++)
        {
            node = node.next;
        }
        Node nodeToDelete = node.next;
        node.next = nodeToDelete.next;
        size--;
    }
}

}

// Problem3.Node

```

```

using System;

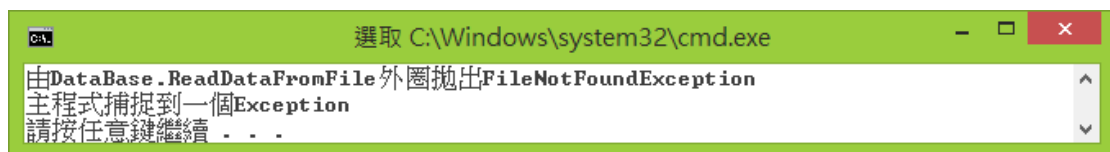
namespace Problem3
{
    public class Node
    {
        public string data = string.Empty;
        public Node next = null;
        public Node(string data, Node next)
        {
            this.data = data;
            this.next = next;
        }
    }
}

```

4. 試寫出以下程式在下列狀況時的主控台螢幕輸出(詞文可以只寫頭尾，中間使用省略號「...」)。注意英文逗點「,」與中文逗點「，」在電腦裡的 Unicode 不同。

(a) (3%) 檔案 **Ci.dat** 尚未建立

答：



(b) (3%) 檔案 **Ci.dat** 已在正確位置，且內容為

3

蘇軾,定風波, 1, 2

三月七日，沙湖道中遇雨，雨具先去，同行皆狼狽，余獨不覺。已而遂晴，故作此詞。

莫聽穿林打葉聲。何妨吟嘯且徐行。竹杖芒鞋輕勝馬。誰怕？一蓑煙雨任平生。

料峭春風吹酒醒。微冷。山頭斜照卻相迎。回首向來蕭瑟處。歸去。也無風雨也無晴。

陳與義,臨江仙,1,2

夜登小閣，憶洛中舊遊。

憶昔午橋橋上飲，坐中多是豪英。長溝流月去無聲。杏花疏影裡，吹笛到天明。

二十餘年如一夢，此身雖在堪驚。閑登小閣看新晴。古今多少事，漁唱起三更。

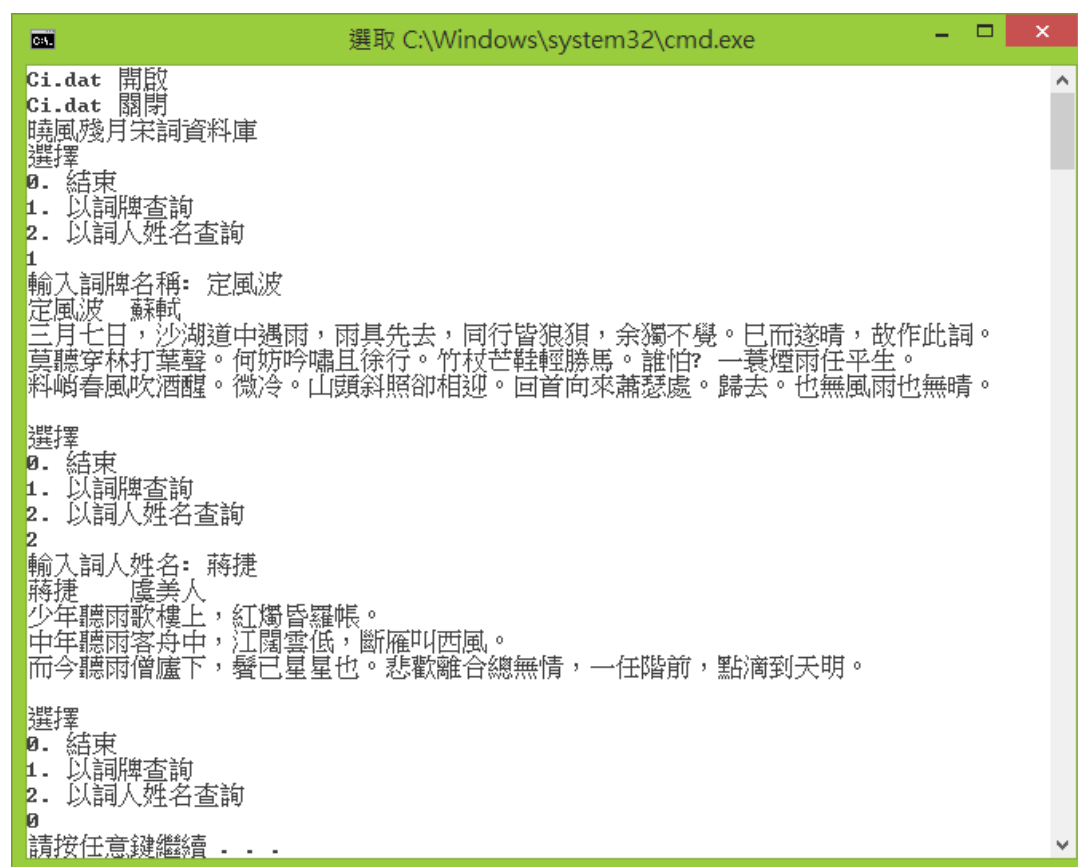
蔣捷,虞美人,0,8

少年聽雨歌樓上，紅燭昏羅帳。

中年聽雨客舟中，江闊雲低，斷雁叫西風。

而今聽雨僧廬下，鬢已星星也。悲歡離合總無情，一任階前，點滴到天明。

答：原題沒有說明使用者的互動過程，假設第一次以詞牌「定風波」搜尋，第二次以「蔣捷」搜尋。其主控台螢幕輸出將為：



```
C:\Windows\system32\cmd.exe
Ci.dat 開啟
Ci.dat 關閉
曉風殘月宋詞資料庫
選擇
0. 結束
1. 以詞牌查詢
2. 以詞人姓名查詢
1
輸入詞牌名稱: 定風波
定風波 蘇軾
三月七日，沙湖道中遇雨，雨具先去，同行皆狼狽，余獨不覺。已而遂晴，故作此詞。
莫聽穿林打葉聲。何妨吟嘯且徐行。竹杖芒鞋輕勝馬。誰怕？一蓑煙雨任平生。
料峭春風吹酒醒。微冷。山頭斜照卻相迎。回首向來蕭瑟處。歸去。也無風雨也無晴。
選擇
0. 結束
1. 以詞牌查詢
2. 以詞人姓名查詢
2
輸入詞人姓名: 蔣捷
蔣捷 虞美人
少年聽雨歌樓上，紅燭昏羅帳。
中年聽雨客舟中，江闊雲低，斷雁叫西風。
而今聽雨僧廬下，鬢已星星也。悲歡離合總無情，一任階前，點滴到天明。
選擇
0. 結束
1. 以詞牌查詢
2. 以詞人姓名查詢
0
請按任意鍵繼續...
```

(c) (3%) 檔案 Ci.dat 已在正確位置，且內容為

3

蘇軾,定風波, 1, 2

三月七日，沙湖道中遇雨，雨具先去，同行皆狼狽，余獨不覺。已而遂晴，故作此詞。

莫聽穿林打葉聲。何妨吟嘯且徐行。竹杖芒鞋輕勝馬。誰怕？一蓑煙雨任平生。

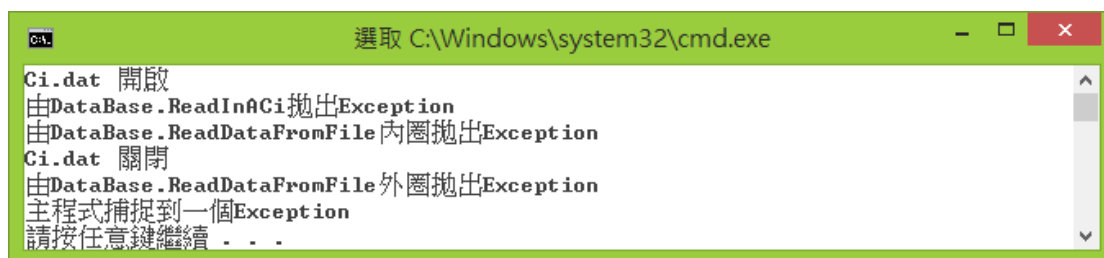
料峭春風吹酒醒。微冷。山頭斜照卻相迎。回首向來蕭瑟處。歸去。也無風雨也無晴。

陳與義,臨江仙,1,2

憶昔午橋橋上飲，坐中多是豪英。長溝流月去無聲。杏花疏影裡，吹笛到天明。
二十餘年如一夢，此身雖在堪驚。閑登小閣看新晴。古今多少事，漁唱起三更。
蔣捷,虞美人,0,3
少年聽雨歌樓上，紅燭昏羅帳。
中年聽雨客舟中，江闊雲低，斷雁叫西風。
而今聽雨僧廬下，鬢已星星也。悲歡離合總無情，一任階前，點滴到天明。

答:

陳與義臨江仙的詞序「夜登小閣，憶洛中舊遊。」不見了，因此程式會把「蔣捷,虞美人,0,3」當成陳與義詞的一部分，接下來蔣捷詞的「少年聽雨歌樓上，紅燭昏羅帳。」無法 Parse 出來新詞的作者，詞牌等內容，產生例外。所以主控台螢幕畫面成為:



(d) (3%) 檔案 Ci.dat 已在正確位置，且內容為

3

蘇軾,定風波, 1, 3

三月七日，沙湖道中遇雨，雨具先去，同行皆狼狽，余獨不覺。已而遂晴，故作此詞。
莫聽穿林打葉聲。何妨吟嘯且徐行。竹杖芒鞋輕勝馬。誰怕？一蓑煙雨任平生。
料峭春風吹酒醒。微冷。山頭斜照卻相迎。回首向來蕭瑟處。歸去。也無風雨也無晴。

陳與義,臨江仙,1,2

夜登小閣，憶洛中舊遊。

憶昔午橋橋上飲，坐中多是豪英。長溝流月去無聲。杏花疏影裡，吹笛到天明。
二十餘年如一夢，此身雖在堪驚。閑登小閣看新晴。古今多少事，漁唱起三更。

蔣捷,虞美人,0,3

少年聽雨歌樓上，紅燭昏羅帳。

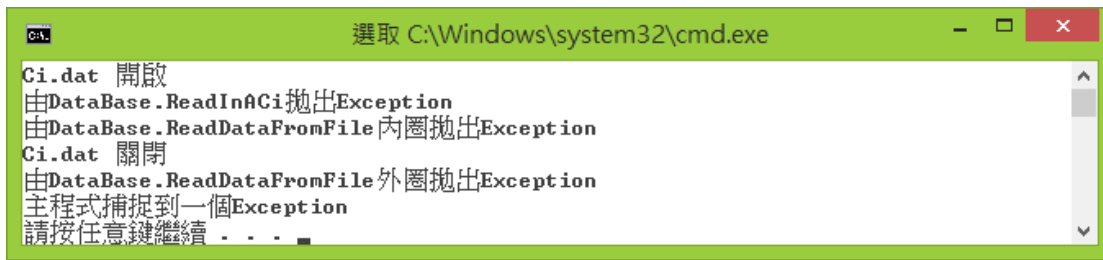
中年聽雨客舟中，江闊雲低，斷雁叫西風。

而今聽雨僧廬下，鬢已星星也。悲歡離合總無情，一任階前，點滴到天明。

答:

情況與上一題相似，只是蘇軾詞多占一行，陳與義詞的作者等資訊就沒有被抓倒

了。主控台螢幕畫面為:



```
// Problem4.Program
using System;

namespace Problem4
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                DataBase db = new DataBase("Ci.dat");
                db.Process();
            }
            catch(Exception e)
            {
                Console.WriteLine("主程式捕捉到一個Exception");
            }
        }
    }
}

// Program4.DataBase
using System;
using System.IO;

namespace Problem4
{
    public class DataBase
    {
```



```

private Ci[] cis = null;
private int nCis = 0;
public DataBase() {}
public DataBase(string fileName)
{
    ReadDataFromFile(fileName);
}

public void Process()
{
    ShowHeading();
    while (true)
    {
        int request = GetRequest();
        if (request == 0) break;
        if (request == 1) ProcessQueryByCipai();
        if (request == 2) ProcessQueryByAuthor();
    }
}

private void ShowHeading()
{
    Console.WriteLine("曉風殘月宋詞資料庫");
}

private int GetRequest()
{
    Console.WriteLine("選擇");
    Console.WriteLine("0. 結束");
    Console.WriteLine("1. 以詞牌查詢");
    Console.WriteLine("2. 以詞人姓名查詢");
    int request = int.Parse(
        Console.ReadLine());
    return request;
}

private void ProcessQueryByCipai()
{

```

```

        Console.WriteLine("輸入詞牌名稱: ");
        string cipai = Console.ReadLine();
        for (int i = 0; i < cis.Length; i++)
        {
            if (cis[i].cipai == cipai)
            {
                Console.WriteLine(
                    cis[i].cipai + "\t" +
                    cis[i].author);
                Console.WriteLine(cis[i].preface);
                Console.WriteLine(cis[i].verse);
            }
        }
        Console.WriteLine();
    }

    private void ProcessQueryByAuthor()
    {
        Console.WriteLine("輸入詞人姓名: ");
        string author = Console.ReadLine();
        for (int i = 0; i < cis.Length; i++)
        {
            if (cis[i].author == author)
            {
                Console.WriteLine(
                    cis[i].author + "\t" +
                    cis[i].cipai);
                Console.WriteLine(cis[i].preface);
                Console.WriteLine(cis[i].verse);
            }
        }
        Console.WriteLine();
    }

    private void ReadDataFromFile(string fileName)
    {
        try
        {

```

```

StreamReader input = new StreamReader(fileName);
Console.WriteLine(fileName + " 開啟");

try
{
    string line = input.ReadLine();
    nCis = int.Parse(line);
    cis = new Ci[nCis];

    int idx = 0;
    while (!input.EndOfStream)
    {
        ReadInACi(input, idx);
        idx++;
    }
}
catch (IOException e)
{
    Console.WriteLine(
        "由DataBase.ReadDataFromFile內圈拋出IOException");
    throw e;
}
catch (Exception e)
{
    Console.WriteLine(
        "由DataBase.ReadDataFromFile內圈拋出Exception");
    throw e;
}
finally
{
    input.Close();
    Console.WriteLine(fileName + " 關閉");
}
}
catch (FileNotFoundException e)
{
    Console.WriteLine(
        "由DataBase.ReadDataFromFile外圈拋出FileNotFoundException");
}

```

```

        throw e;
    }
    catch (Exception e)
    {
        Console.WriteLine(
            "由DataBase.ReadDataFromFile外圈拋出Exception");
        throw e;
    }
}

private void ReadInACi(StreamReader input, int idx)
{
    // 讀入第idx首詞
    char[] delimiters = new char[] { ' ', ',', '.', ' ' };
    string[] str = new string[4];
    string line = string.Empty;
    int nLinesPreface = 0;
    int nLinesVerse = 0;

    try
    {
        line = input.ReadLine();
        str = line.Split(delimiters);
        cis[idx].author = str[0];
        cis[idx].cipai = str[1];
        nLinesPreface = int.Parse(str[2]);
        nLinesVerse = int.Parse(str[3]);

        cis[idx].preface = string.Empty;
        for (int i = 0; i < nLinesPreface; i++)
        {
            string prefaceLine = input.ReadLine() + "\n";
            cis[idx].preface += prefaceLine;
        }

        cis[idx].verse = string.Empty;
        for (int i = 0; i < nLinesVerse; i++)
        {

```

```

        string verseLine = input.ReadLine() + "\n";
        cis[idx].verse += verseLine;
    }
}
catch(Exception e)
{
    Console.WriteLine("由DataBase.ReadInACi拋出Exception");
    throw e;
}
}
}

// Problem4.Ci
using System;

namespace Problem4
{
    public struct Ci // 詞
    {
        public string author; // 詞人
        public string cipai; // 詞牌
        public string preface; // 詞序
        public string verse; // 詞文

        public Ci(string author, string cipai, string preface,
            string verse)
        {
            this.author = author;
            this.cipai = cipai;
            this.preface = preface;
            this.verse = verse;
        }
    }
}

```

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：水平拋體動畫

建立如圖 4 之圖形使用介面，按下「發射」按鈕 (**button1**)，即於主視窗出現一個紅色橢圓緩慢以水平拋體軌跡下降至主視窗邊界後消失。紅色橢圓運動時，「發射」按鈕變成灰色，表示 **button1** 的屬性 **Enable** 成為 **false**；當紅色橢圓消失，**button1** 之 **Enable** 重設為 **true**，「發射」按鈕恢復原有顏色。

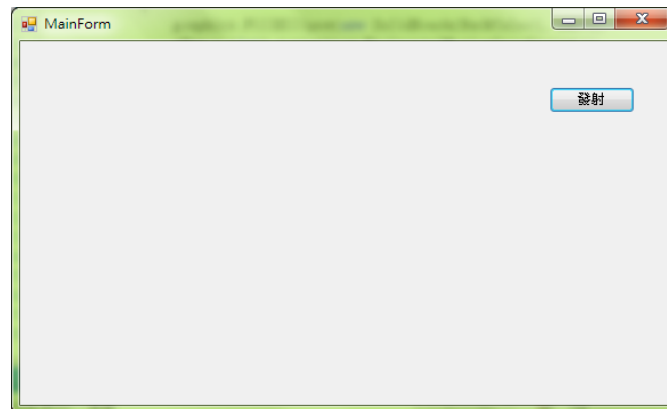


圖 4. 程式開始執行時之視窗畫面

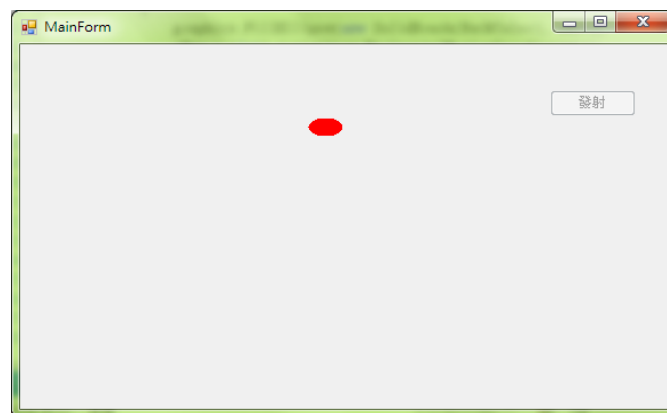


圖 5. 按下「發射」按鈕後某時間截取之視窗畫面

```
// 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 Graphics graphics = null;
        private Trajectory trajectory = null;
        private int t = 0;
        private int x0 = 0;
        private int y0 = 0;
        private double v0 = 0.0;
        private double g = 0.0;
        private int xNew = 0;
        private int yNew = 0;
        private int w = 0;
        private int h = 0;
        private Rectangle rect;
        //*****

        public MainForm()
        {
            InitializeComponent();
            //*****
            w = 30;
            h = 16;
            x0 = w / 2;
            y0 = h / 2;
            v0 = 7.0;
            g = 0.1;
            trajectory = new Trajectory(x0, y0, v0, g);
            graphics = this.CreateGraphics();
            //*****
        }

        private void button1_Click(object sender, EventArgs e)

```

```

{
    //*****
    timer1.Start(); // timer1為由工具箱拖曳進主視窗的計時器
        // timer1.Start()表示timer1開始計時

    t = 0;
    rect = new Rectangle(x0 - w / 2, y0 - h / 2, w, h);
        // 紅色橢圓的外接矩形
        // 建構式第一，二個引數為矩形左上角pixel座標
        // 第三，四個引數為矩形的寬(x方向)和高(y方向)

    graphics.FillEllipse(new SolidBrush(Color.Red), rect);
    button1.Enabled = false;
    //*****
}

private void timer1_Tick(object sender, EventArgs e)
{
    //*****
    ++t;
    if(t % 2 == 0)
    {
        // 請依照以下說明及本題其他程式碼完成此處應有的程式敘述 (6%)
        // 1. 選用new SolidBrush(BackColor)將原來橢圓區域改塗成背景色
        // 亦即使其消失
        // 2. 以類別Trajectory中的HorizontalMotionCoordinate及
        // VerticalMotionCoordinate計算新橢圓中心位置座標
        // xNew及yNew
        // 3. 如果xNew和yNew決定的新橢圓外接矩形會觸及視窗邊界，則呼叫
        // timer1.Stop()停止計時，並使button1重新被Enable
        // 4. 如果xNew和yNew決定之橢圓外接矩形仍在視窗有效區域內
        // 則以紅色SolidBrush繪出xNew和yNew決定的新橢圓，製造出
        // 動畫效果

        // 答：
        graphics.FillEllipse(new SolidBrush(BackColor), rect);
        xNew = (int) trajectory.HorizontalMotionCoordinate(t);
        yNew = (int) trajectory.VerticalMotionCoordinate(t);
        if(TouchBoundaries(xNew, yNew))

```



```

        {
            button1.Enabled = true;
            timer1.Stop();
        }
        else
        {
            rect = new Rectangle(xNew-w/2, yNew-h/2, w, h);
            graphics.FillEllipse(new SolidBrush(Color.Red),
                rect);
        }
    }
    //*****

}

//*****
private bool TouchBoundaries(int x, int y)
{
    bool touchBoundaries = (x - w/2 < 0) ||
                            (x + w/2 > this.Width) ||
                            (y - h/2 < 0) ||
                            (y + h/2 > this.Height);

    return touchBoundaries;
}
//*****
}

// Problem5.Trajectory
using System;

namespace Problem5
{
    public class Trajectory
    {
        private double x0 = 0.0;
        private double y0 = 0.0;
        private double v0 = 0.0;
        private double g = 0.0;
    }
}

```

```

private double x = 0.0;
private double y = 0.0;

public Trajectory(double x0, double y0, double v0, double g)
{
    this.x0 = x0;
    this.y0 = y0;
    this.v0 = v0;
    this.g = g;
    x = x0;
    y = y0;
}

public double HorizontalMotionCoordinate(double t)
{
    x = x0 + v0*t;
    return x;
}

public double VerticalMotionCoordinate(double t)
{
    y = y0 + 0.5 * g * t * t;
    return y;
}
}
}

```

6. 認知心理學(Cognitive Psychology)探討認知現象及行為背後之心智處理過程，如大腦思維、學習、決策、推理、動機、審美、情緒產生與控制等。認知心理學的研究一般都需要藉由實驗來驗證各種心智處理方法的假說與模型解釋：實驗通常包含一件或數件要求受試者(人類或靈長類如恆河猴、黑猩猩)完成的 **task**，藉由記錄分析受試者進行 **task** 時表現的之行為選擇及/或物理生理反應，如反應時間 **reaction time**、皮膚導電度反應 **skin conductivity response (SCR)**、腦部功能性磁共振造影 **functional Magnetic Resonance Imaging (fMRI)**、相關神經細胞群激發程度等等，檢視理論假說、數值方程式模型、非數值定性模型(通常以方塊和箭頭表示神經系統內不帶數值的訊號流程，因此又稱 **box-and-arrow models**)等，所預測的行為或量測訊號是否與實驗結果有相當的一致性。

在學習與決策方面的認知心理學實驗中，應用 Iowa Gambling Task (IGT) 的實驗可說是經典之作。此一 task 係由美國 University of Iowa 的四位研究人員於 1994 年提出，用以模擬人類生活中的決策行為，探討常人與腦傷病人的差異。進一步的說明可以參考 https://en.wikipedia.org/wiki/Iowa_gambling_task 。

在 IGT 中，受試者由固定起始虛擬貨幣金額開始，不斷由四疊卡片擇一翻牌；牌面註明受試者可以得到多少獎賞及懲罰之虛擬貨幣金額。受試者在 task 中的目標為贏取最高金額之虛擬貨幣，但因每一疊卡片內出現懲罰牌卡的罰金數額不同，所以長遠看來，可能會有短期收益較低，但罰款金額也少，使長期翻取淨總金額變高的「好牌疊」；也有可能牌面獎賞較高，但其罰款金額亦甚高，使長期淨總金額變低的「壞牌疊」。實驗顯示：常人於 40 或 50 次選擇之後，會逐漸固著於選擇「好牌疊」，而大腦眼窩前額皮質(orbitofrontal cortex)受損病人卻常繼續選擇「壞牌疊」，腹內側前額皮質(ventromedial prefrontal cortex)受損病人則會不顧長遠利益，而經常選擇短期收益最高的牌疊。

以往這類的認知心理學實驗進行時，實驗者需重複對不同受試者說明相同之實驗過程及細節(experiment instruction)，並且隨機安排牌卡出現順序與手動計算得分，工作相當繁瑣無趣。近年來，這些工作多半已由電腦代勞，而且出現心理學實驗安排專用的高階程式語言，例如開放原始碼 PEBL (Psychology Experiment Building Language)，請參閱 [https://en.wikipedia.org/wiki/PEBL_\(software\)](https://en.wikipedia.org/wiki/PEBL_(software)) 、
<https://www.youtube.com/watch?v=7Z8TU20Ryyg> 、
<http://pebl.sourceforge.net/> 。

此外，也有許多研究，在符合研究主題之要求下，可將實驗過程放在網站上，受試者可以選擇自己方便的時間參與實驗，樣本數也會較安排定時定點之實驗為多；甚且可以收集到跨背景例如外國人的受試結果，前提當然是受試者在受試前，不得與他人討論實驗詳情及作答策略。

由於存在需求，遂有公司成立，提供上述實驗室內與網路上心理學實驗的軟體及其衍生的服務：例如 Millisecond (<https://www.millisecond.com/about/about.aspx>)，主力產品 Inquisit Lab 及 Inquisit Web，分別為實驗室及網路用軟體。除 Inquisit Lab 可免費試用 30 天外，收費相當高昂：Inquisit Lab 首次購買價格為第一部安裝電腦美金 450 元，隨後每套 license 美金 150 元；升級價格為第一部安裝電腦美金 250 元，隨後每套 license 美金 100 元。Inquisit Web 則為租賃方式販售，入門價為第一部電腦安裝租用兩個月美金 395 元，第二部電腦之後，每機安裝兩個月的 license fee 為 295 美元。Inquisit Lab 及 Inquisit Web 對於大量長期或使用單位授權則另有若干優惠。在此一高價位之下，全世界仍有數以百計的心理學研究與社會

工作有關單位成為 **Millisecond** 客戶，顯見其重要性，以及規模足夠之市場存在。

本題假定研究團隊經費不足以購買或租用 **Inquisite Lab** 與 **Inquisite Web**，在時程壓力下，亦無人力及時間安裝與熟悉 **PEBL**(這也許不是一個好決定)；因此希望團隊中唯一具備程式開發能力的你（當然是因為只有你修了一學期鄭老師和郭助教開的「通識計算機程式設計」），自力撰寫一個程式，幫助本團隊中的認知心理學者，利用計算機安排 **IGT** 牌疊、與受試者互動進行實驗、同時統計各受試者積分，以便協助了解人腦的認知決策過程。此一程式的主要流程虛擬碼假設為：

1. 建立四個牌疊
2. 顯示參加實驗訊息
3. 受試者輸入四位數字之代號
4. **while**(選擇次數未滿)
 - {
 - 4.1 對受試者顯示提示
 - 4.2 紀錄受試者的選擇
 - 4.3 顯示受試者得到的本次獎賞及懲罰金額、本次選擇淨所得與累計淨總金額
 - }
5. 顯示實驗結束訊息
6. 顯示受試者於全部選擇中，「好牌疊」及「壞牌疊」的被選擇次數比例
7. 顯示受試者於某次選擇後到結束，「好牌疊」及「壞牌疊」的被選擇次數比例

依據 A. Bechara, H. Damasio, D. Tranel and A.R. Damasio, "The Iowa Gambling Task and the somatic marker hypothesis: some questions and answers," *TRENDS in Cognitive Sciences*, Vol.9 No.4 , April 2005, pp. 159 – 162 (http://waldron.stanford.edu/~jlm/papers/BecharaEtAl05_TiCS.pdf) 論文中的 Box1 說明：受試者一共抽取 100 張牌(所以設定四個牌疊各有 100 張牌)，牌疊性質如圖 6 所示。

The Iowa Gambling Task				
	"Bad" decks		"Good" decks	
	A	B	C	D
Gain per card	\$100	\$100	\$50	\$50
Loss per 10 cards	\$1250	\$1250	\$250	\$250
Net per 10 cards	-\$250	-\$250	+\$250	+\$250

TRENDS in Cognitive Sciences

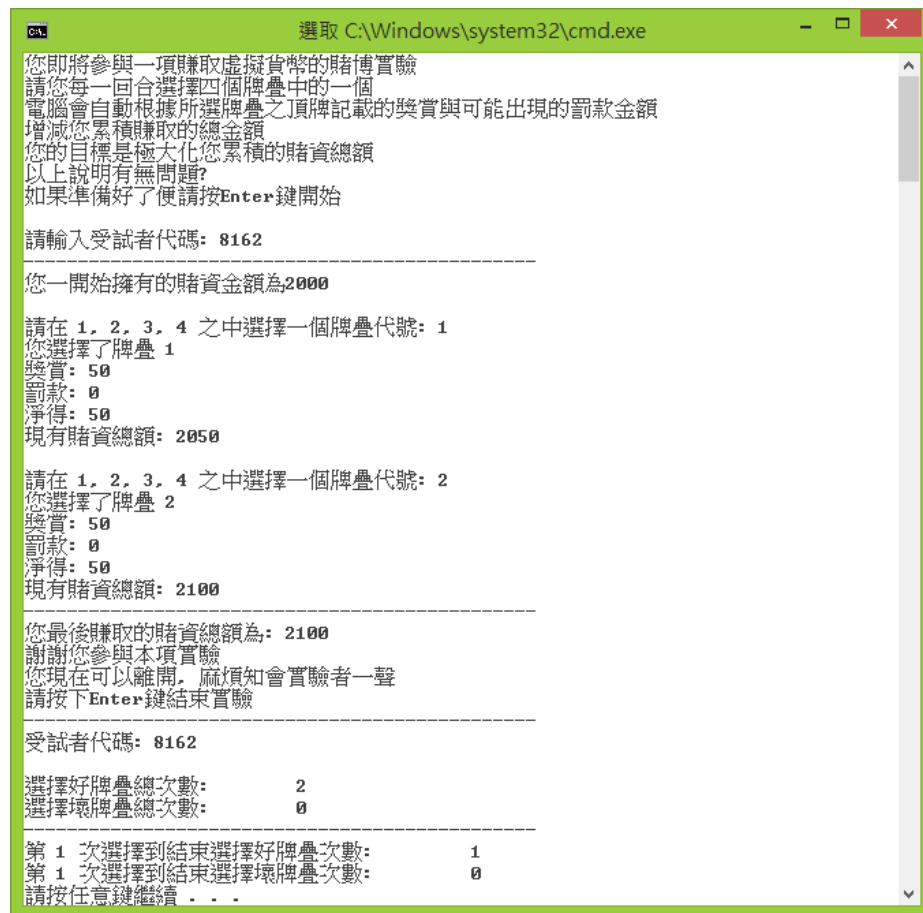
圖 6. Iowa Gambling Task 的牌疊性質

亦即，四個牌疊分為兩組：

牌疊 A 與牌疊 B: 每張牌固定獎賞 100 元，每十張牌有一張另處罰 1250 元，使每十張牌的期望淨值為-250 元（「壞牌疊」）。

牌疊 C 與牌疊 D: 每張牌固定獎賞 50 元，每十張牌有一張另處罰 250 元，使每十張牌的期望淨值為 250 元（「好牌疊」）。

程式執行的某一場景如圖 7。



```
您即將參與一項賺取虛擬貨幣的賭博實驗
請您每一回合選擇四個牌疊中的一個
電腦會自動根據所選牌疊之頂牌記載的獎賞與可能出現的罰款金額
增減您累積賺取的總金額
您的目標是極大化您累積的賭資總額
以上說明有無問題?
如果準備好了便請按Enter鍵開始

請輸入受試者代碼: 8162

-----
您一開始擁有的賭資金額為2000

請在 1, 2, 3, 4 之中選擇一個牌疊代號: 1
您選擇了牌疊 1
獎賞: 50
罰款: 0
淨得: 50
現有賭資總額: 2050

請在 1, 2, 3, 4 之中選擇一個牌疊代號: 2
您選擇了牌疊 2
獎賞: 50
罰款: 0
淨得: 50
現有賭資總額: 2100

-----
您最後賺取的賭資總額為: 2100
謝謝您參與本項實驗
您現在可以離開, 麻煩知會實驗者一聲
請按下Enter鍵結束實驗

-----
受試者代碼: 8162

選擇好牌疊總次數:      2
選擇壞牌疊總次數:      0

-----
第 1 次選擇到結束選擇好牌疊次數:      1
第 1 次選擇到結束選擇壞牌疊次數:      0
請按任意鍵繼續 . . .
```

圖 7. Iowa Gambling Task 實驗控制程式執行畫面範例

以上述之說明參照以下提示撰寫程式，不必處理「例外」(exception)。固定「好牌疊」及「壞牌疊」位置者(例如，固定令第一、二牌疊為「好牌疊」，第三、四牌疊為「壞牌疊」)，最高得 22 分。讓電腦隨機決定好、壞各兩牌疊者(亦即，這次執行程式，可能第一、二牌疊為「好牌疊」，第三、四牌疊為「壞牌疊」；下次執行時，可能變成第一、三牌疊為「好牌疊」，第二、四牌疊為「壞牌疊」)最高得 25 分。雖然如此，鼓勵你先寫固定牌疊的版本，完成後再依所剩考試時間長短決定是否撰寫隨機牌疊的版本。(25%)

提示：

1). 可以直接使用已經完成的結構 Card 與類別 Deck 程式如下(不用抄內容)：

```
public struct Card
{
    public int reward;
    public int penalty;
    public Card(int reward, int penalty)
    {
        this.reward = reward;
        this.penalty = penalty;
    }
}

public class Deck
{
    private int nCards = 0;
    private int penaltyPeriod = 0;
    private Card[] cards = null;
    private int top = 0;
    private Random rand = new Random();

    public Deck(int nCards, int penaltyPeriod, int reward, int penalty)
    {
        this.nCards = nCards;
        this.penaltyPeriod = penaltyPeriod;
        top = 0;
        cards = new Card[nCards];
        int q = nCards / penaltyPeriod;
        int iCard = 0;
        int i;
        for (int k = 0; k < q; k++ )
        {
            int r = rand.Next(penaltyPeriod);
            for (i = 0; i < penaltyPeriod; i++)
            {
                cards[iCard] = (i == r) ?
                    new Card(reward, penalty) : new Card(reward, 0);
                iCard++;
            }
        }
    }
}
```

```

    }
}
for(i = 0; i < nCards % penaltyPeriod; i++)
{
    cards[iCard] = new Card(reward, 0);
    iCard++;
}
}

public Card Top()
{
    Card topCard = cards[top];
    top++;
    return topCard;
}
}

```

2) 實驗開始顯示訊息之宣告如下，可任意放置於你認為適合之處。此一宣告在你的程式中記為 `const string INSTRUCTIONS = ". . . "`；即可，不用再抄一遍其內容。

```

const string INSTRUCTIONS = "您即將參與一項賺取虛擬貨幣的賭博實驗\n" +
    "請您每一回合選擇四個牌疊中選擇一個翻開頂牌\n" +
    "電腦會自動根據頂牌記載的獎賞與可能出現的罰款金額\n" +
    "增減您累積賺取的總金額\n" +
    "您的目標是極大化您累積的賭資總額\n" +
    "以上說明有無問題? \n" +
    "如果準備好了便請按 Enter 鍵開始\n";

```

3). 實驗中對受試者顯示的提示如下，可任意放置於你認為適合之處。此一宣告在你的程式中記為 `const string PROMPT = ". . . "`；即可，不用再抄一遍其內容。

```

const string PROMPT = "請在 1, 2, 3, 4 之中選擇一個牌疊代號: ";

```

4). 實驗結束時顯示的結束訊息如下，可任意放置於你認為適合之處。此一宣告在你的程式中記為 `const string ENDING_MESSAGE = "... "`；即可，不用再抄一遍其內容。

```

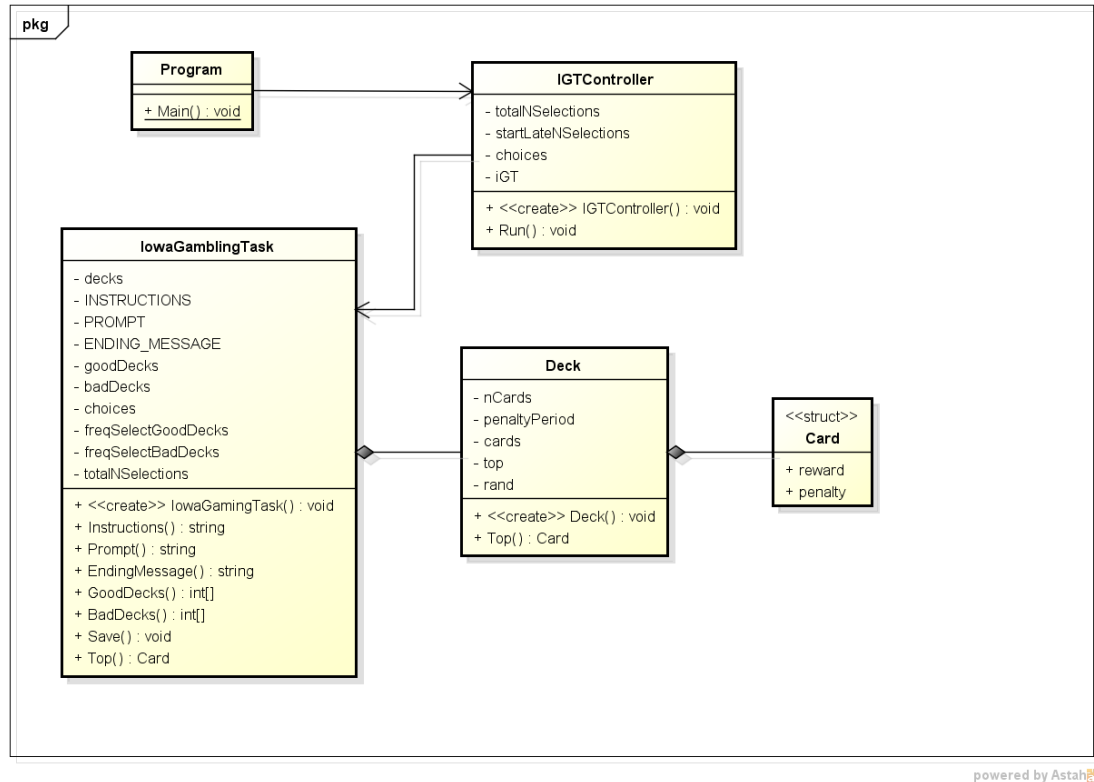
const string ENDING_MESSAGE = "謝謝您參與本項實驗\n" +
    "您現在可以離開，麻煩知會實驗者一聲\n" +

```

"請按下 Enter 鍵結束實驗 \n"

答：

以 UML 思考，繪出類別圖如下：



using System;

namespace Problem6

```
{
    class Program
    {
        static void Main(string[] args)
        {
            const int TOTAL_N_SELECTIONS = 2;
            const int START_LATE_N_SELECTIONS = 1;
            const int N_CARDS = 10;
            const int PENALTY_PERIOD = 5;
            int[] rewards = { 100, 50 };
            int[] penalties = { 1250, 250 };
            IGTCController iGTController = new
```



```

        IGTCController(TOTAL_N_SELECTIONS,
            START_LATE_N_SELECTIONS, N_CARDS, PENALTY_PERIOD,
            rewards, penalties);
        const int INITIAL_LOAN = 2000;
        iGTController.Run(INITIAL_LOAN);
    }
}

using System;

namespace Problem6
{
    public class IGTCController
    {
        private int totalNSelections = 0;
        private int startLateNSelections = 0;
        private int[] choices = null;
        private IowaGamblingTask iGT = null;

        public IGTCController(int totalNSelections,
            int startLateNSelections,
            int nCards, int penaltyPeriod, int[] rewards,
            int[] penalties)
        {
            this.totalNSelections = totalNSelections;
            this.startLateNSelections = startLateNSelections;
            choices = new int[totalNSelections];
            iGT = new IowaGamblingTask(nCards, penaltyPeriod, rewards,
                penalties);
        }

        public void Run(int initialLoan)
        {
            Display(iGT.Instructions);
            Console.ReadLine();
            Display("請輸入受試者代碼: ");
            string subjectID = Console.ReadLine();

```

```

Console.WriteLine(
    "-----");
int reward = 0;
int penalty = 0;
Card chosenCard;
Console.WriteLine("您一開始擁有的賭資金額為" + initialLoan);
int total = initialLoan;
for(int n = 0; n < totalNSelections; n++)
{
    Console.WriteLine();
    choices[n] = InputChoice(iGT.Prompt);
    chosenCard = iGT.Top(choices[n]);
    reward = chosenCard.reward;
    penalty = chosenCard.penalty;
    total += (reward - penalty);
    DisplayResults(choices[n], reward, penalty, total);
}
iGT.Save(choices);

Console.WriteLine(
    "-----");
Console.WriteLine("您最後賺取的賭資總額為: " + total);
Display(iGT.EndingMessage);

Console.WriteLine(
    "-----");
Console.WriteLine("受試者代碼: " + subjectID);
Console.WriteLine();
Console.WriteLine("選擇好牌疊總次數: \t " +
    iGT.FreqSelectGoodDecks);
Console.WriteLine("選擇壞牌疊總次數: \t " +
    iGT.FreqSelectBadDecks);

Console.WriteLine(
    "-----");
Console.WriteLine("第 {0} 次選擇到結束選擇好牌疊次數: \t {1} ",
    startLateNSelections,

```

```

        iGT.NLateFreqSelectGoodDecks(startLateNSelections));
        Console.WriteLine("第 {0} 次選擇到結束選擇壞牌疊次數: \t {1}",
            startLateNSelections,
            iGT.NLateFreqSelectBadDecks(startLateNSelections));
    }

    private void Display(string message)
    {
        Console.Write(message);
    }

    private int InputChoice(string prompt)
    {
        Display(prompt);
        int choice = int.Parse(Console.ReadLine());
        return choice - 1; // 牌疊內部代號為0, 1, 2, 3
    }

    private void DisplayResults(int choice, int reward, int penalty,
        int total)
    {
        Console.WriteLine("您選擇了牌疊 " + (choice+1));
        //螢幕上的牌疊代碼是 1, 2, 3, 4

        Console.WriteLine("獎賞: " + reward);
        Console.WriteLine("罰款: " + penalty);
        Console.WriteLine("淨得: " + (reward - penalty));
        Console.WriteLine("現有賭資總額: " + total);
    }
}

}

using System;

namespace Problem6
{
    public class IowaGamblingTask
    {

```

```

private Deck[] decks = new Deck[4];
private int[] arrangement = new int[4];
private int[] goodDecks = new int[2];
private int[] badDecks = new int[2];
private int[] choices = null;
private int totalNSelections = 0;
private int freqSelectGoodDecks = 0;
private int freqSelectBadDecks = 0;
private const string INSTRUCTIONS =
    "您即將參與一項賺取虛擬貨幣的賭博實驗\n" +
    "請您每一回合選擇四個牌疊中的一個\n" +
    "電腦會自動根據所選牌疊之頂牌記載的獎賞與可能出現的罰款金額\n" +
    "增減您累積賺取的總金額\n" +
    "您的目標是極大化您累積的賭資總額\n" +
    "以上說明有無問題? \n" +
    "如果準備好了便請按Enter鍵開始\n";
private const string PROMPT =
    "請在 1, 2, 3, 4 之中選擇一個牌疊代號: ";
private const string ENDING_MESSAGE =
    "謝謝您參與本項實驗\n" +
    "您現在可以離開, 麻煩知會實驗者一聲\n" +
    "請按下Enter鍵結束實驗 \n";

public IowaGamblingTask(int nCards, int penaltyPeriod,
    int[] rewards, int[] penalties)
{
    ShuffleDecks();

    // generate decks
    for (int i = 0; i < 4; i++)
    {
        int k = i / 2;
        decks[arrangement[i]] = new Deck(nCards, penaltyPeriod,
            rewards[k], penalties[k]);
    }

    SeparateGoodAndBadDecks(penaltyPeriod, rewards, penalties);
}

```

```

private void ShuffleDecks()
{
    Random rand = new Random();
    bool again = true;
    for (int i = 0; i < 4; i++)
    {
        int r = 0;
        again = true;
        while (again)
        {
            again = false;
            r = rand.Next(4);
            for (int j = 0; j < i; ++j)
            {
                if (arrangement[j] == r)
                {
                    again = true;
                    break;
                }
            }
        }
        arrangement[i] = r;
    }
}

private void SeparateGoodAndBadDecks(int penaltyPeriod,
    int[] reward, int[] penalty)
{
    goodDecks = new int[2];
    badDecks = new int[2];
    if (ExpectedDeckValue(penaltyPeriod, reward[0], penalty[0]) >
        ExpectedDeckValue(penaltyPeriod, reward[1], penalty[1]))
    {
        goodDecks[0] = arrangement[0];
        goodDecks[1] = arrangement[1];
        badDecks[0] = arrangement[2];
        badDecks[1] = arrangement[3];
    }
}

```

```

    }
    else
    {
        badDecks[0] = arrangement[0];
        badDecks[1] = arrangement[1];
        goodDecks[0] = arrangement[2];
        goodDecks[1] = arrangement[3];
    }
}

public string Instructions
{
    get { return INSTRUCTIONS; }
}

public string Prompt
{
    get { return PROMPT; }
}

public string EndingMessage
{
    get { return ENDING_MESSAGE; }
}

public int[] GoodDecks()
{
    return goodDecks;
}

public int[] BadDecks()
{
    return badDecks;
}

public Card Top(int k)
{
    return decks[k].Top();
}

```

```

    }

    private int ExpectedDeckValue(int penaltyPeriod, int reward,
        int penalty)
    {
        return (reward * penaltyPeriod - penalty);
    }

    public void Save(int[] choices)
    {
        totalNSelections = choices.Length;
        this.choices = new int[totalNSelections];
        Array.Copy(choices, this.choices, totalNSelections);
        freqSelectGoodDecks = NGoodDecksSelections(0);
        freqSelectBadDecks = NBadDeckSelections(0);
    }

    public int FreqSelectGoodDecks
    {
        get { return freqSelectGoodDecks; }
    }

    public int FreqSelectBadDecks
    {
        get { return freqSelectBadDecks; }
    }

    public int NLateFreqSelectGoodDecks(int startingNLateSelections)
    {
        return NGoodDecksSelections(startingNLateSelections);
    }

    public int NLateFreqSelectBadDecks(int startingNLateSelections)
    {
        return NBadDeckSelections(startingNLateSelections);
    }

    private int NGoodDecksSelections(int nStart)

```

```

    {
        int choice = 0;
        int nSelectGoodDecks = 0;
        for(int n = nStart; n < totalNSelections; n++)
        {
            choice = choices[n];
            if(choice == goodDecks[0] || choice == goodDecks[1])
                nSelectGoodDecks++;
        }
        return nSelectGoodDecks;
    }

private int NBadDeckSelections(int nStart)
{
    int choice = 0;
    int nSelectBadDecks = 0;
    for(int n = nStart; n < totalNSelections; n++)
    {
        choice = choices[n];
        if(choice == badDecks[0] || choice == badDecks[1])
            nSelectBadDecks++;
    }
    return nSelectBadDecks;
}

}
}

```