程式邏輯、程式設計與 C#的基礎 HW4

**Part1 問答題**

1. 第八章 第二題

   ToUpper() 結果為 VISUAL C# 程式設計範例教本

   Substring(2,4) 結果為 sual

   IndexOf(“程式”) 結果為 10

2. 第八章 第六題

   (一)排序: 透過指定條件將一組數據進行排序，以整數陣列來說，我們可以按照大小由小至大、按照大小由大至小、按照位數由大至小...等等。

   (二)搜尋: 在一群資料中，去搜尋指定目標值，查看是否存在。補充:回傳值可以以 bool 來表示有沒有找到，或是透過 int 回傳目標值的索引。而搜尋的方法依據資料可以分為沒有排序的資料及已經排序的資料兩種。

3. 第九章 第一題

   簡單的說，傳統應用程式開發是將”資料”與”操作”分開來思考，透過結構化分析，著重於解決問題的函數，以 C 語言程式為例，是以函數為區塊。

   而物件導向開發則是將”資料”與”操作”，合併來思考形成物件，如同真實世界的物件的模型，以 Java、C#、C++等語言程式微例，是以類別為區塊。

4. 第九章 第六題

   (1)private 只供類別內使用，其中類別成員沒有宣告修飾子則預設為 private。(2)protected 供類別內使用及即子類別使用。

   (3)public 供類別內及所有套件取用，是存取範圍最大同時最寬鬆的權限。

   (4)三者應用上簡易差異與說明 :private 與 public 是最常用的，當我們需要隱藏成員，可以使用 private，例如一個 Circle 類別，為了防止隨意修改半徑，而導致邏輯上的不合理，或實際結果的錯誤，可以將該變數權限設定為 private，以便達到保護作用，只供使用者採用方法的方式進行修改與存取，當使用者想要改半徑都會透過該方法進行檢測;當我們要對外的使用介面，可以採用 public，但程式設計師也要有確保出錯的可能性喔!至於 protected 相較前兩者除非是有特別上的需求，需要繼承才可用則可以採用 protected。

   (5)權限為 private 的方法，又稱工具方法，意為只供類別內的方法成員做呼叫。

## Part2 實作題

### 1. 第八章 第二題

```
int[] arr = new int[5];
for(int i = 0;i < arr.Length; i++)
{
    arr[i] = new Random().Next(1,201);
}
Array.Sort(arr);
for(int i = 0; i < arr.Length; i++)
{
    Console.WriteLine(arr[i]);
}
```

### 2. 第八章 第四題

```
namespace WinFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        int arrMin(int[] arr)
        {
            return arr[0];
        }
        int arrMax(int[] arr)
        {
            return arr[arr.Length-1];
        }
        private void okbtn_Click(object sender, EventArgs e)
        {
            TextBox[] txt_arr = this.Controls.OfType<TextBox>().ToArray();
            int[]  arr = new int[txt_arr.Length];
            for(int i = 0; i < txt_arr.Length; i++)
                arr[i] = Convert.ToInt32(txt_arr[i].Text);
            Array.Sort(arr);
```

```
            minValue.Text = "最小值: " + arrMin(arr);
            maxValue.Text = "最大值: " + arrMax(arr);
        }
    }
}
```

### 3. 第九章 第二題

```
public class Box
{
    double Width, Height, Length;
    Box(double width, double height, double length)
    {
        this.Width = width;
        this.Height = height;
        this.Length = length;
    }
    double Volume()
    {
        return this.Width * this.Height * this.Length;
    }
    double Area()
    {
        return this.Width * this.Length * 6 ;
    }
}
```

| **Box** |
| --- |
| +Width: double<br>+Height: double<br>+Length: double |
| +Volume(): double<br>+Area(): double<br>+Box(width: double, height: double, length: double) |

## 4. 第九章 第四題

```
public class PhoneList
{
    public string? HomePhone;      //住家電話
    public string? BusinessPhone; //公司電話
    public string? CellPhone;      //手機電話
    public PhoneList(string? homePhone, string? businessPhone, string? cellPhone)
    {
        this.HomePhone = homePhone;
        this.BusinessPhone = businessPhone;
        this.CellPhone = cellPhone;
    }
}
public class Cards
{
    public string? Name; //姓名
    public string? Occupation; //職業
    public byte? Age; //年齡
    public PhoneList Phone; //電話
    public string? Email; //電子郵件
    public Cards() : this(null, null, null, null, null, null, null) { }
    public Cards(string? name , string? occupation, byte? age, string? homePhone,
                 string? businessPhone, string? cellPhone,  string? email)
    {
        this.Name = name;
        this.Occupation = occupation;
        this.Age = age;
        Phone = new PhoneList(homePhone, businessPhone, cellPhone);
        this.Email = email;
    }
    public string GetCard()
    {
        return $"姓名: {Name}\n 職位: {Occupation}\n 年齡: {Age}\n 住家電話: {Phone.HomePhone}\n" +
               $"公司電話: { Phone.BusinessPhone}\n 手機電話: { Phone.CellPhone}";
    }
}
```