高雄班

- 元件設計 24hrs
  - 範例檔案 OOP.Solution
  - 宣告 class Demo01.DeclareClass
    - 在主控台應用程式裡宣告類別
      - public class Member
        - 欄位
          - 欄位的宣告方式 — 宣告 string Address
          - 通常宣告為 private or protected
            - public 才能被外界叫用
            - private 或沒有指定,視為 private, 只有類別本身可以叫用
            - protected只有類別本身或子類別才能叫用
          - 練習如何在外界叫用欄位
          - 將欄位宣告成 private — 確認外界無法叫用
        - 屬性
          - 通常取名詞, 例如:高度, 顏色, 尺寸, 姓名, 地址, Email, ...
          - 通常宣告為 public
          - 宣告技巧 — 輸入 'prop' 再按二次 tab
          - 屬性不同宣告內容
            - int Height{get;set;} — 可讀,可寫
            - int Age{get;} — 對外界而言是唯讀, 但類別裡仍然可以有機會修改它 — 如果您針對 get有寫程式傳回私有欄位,那麼在類別裡只要修改欄位值,就能達到修改屬性的效果
            - int Age{get; private set;} — 可以在類別裡修改屬性值
            - int Age { get; protected set;} — 此類別或它的子類別可以修改屬性值
            - int Age{get; init} —
              - 對外是唯讀, 類別只能在建構函數裡指定值, 之後就不允許修改屬性值
              - C#9的新語法
          - Name, Email , Height, DateOfBirth, Gender — 設計各屬性的型別
          - 兩種宣告 property 的寫法
        - 欄位與屬性的差異?
          - 請參考補充教材"屬性.doc"
          - 範例: Demo01.PropertyXX
        - 方法
          - public 才能被外界叫用
          - private 或沒有指定,視為 private, 只有類別本身可以叫用
          - 通常取動詞, 例如: Save, Delete, Search, LoadFile, Authenticate, IsEven, ...
          - string Greet(string greeting)
            - greeting 和 greet 的意思都是「問候，打招呼」。不過，greeting 是名詞，而 greet 是動詞
            - 傳回 'Hello, Allen'
            - 如果 Name 沒有值, 傳回 'Hello'
      - class, property, member 的命名規則
        - 不能用數字開頭
        - 習慣是開頭大寫
      - 欄位的命名規則
        - 習慣是底線開頭
        - 通常是宣告成 private
        - 由於等同於變數, 習慣用小寫開頭
    - NameSpace
  - 練習 — Exec01.DeclareClass
    - class EStore — method
      - int CalcInclusivePrice(int price, double taxRate=0.05)
        - 計算含稅金額
        - 傳入參數
          - 未稅金額
          - 營業稅率=0.05
      - method上的註解
        - What
        - Why
        - How
    - class Math — methods
      - bool IsPrime(int number)
        - 判斷 number 是否為質數
        - 若小於 2 ,傳回 false
        - 若大於等於 2, 傳回 number 是否為質數
      - int RoundToInt(double value) — 將 value 四捨五入並傳回 int 型別
  - 建構函數 — Demo01.Constructor
    - class Member
      - Properties — Name, Email
      - 如果沒有寫建構函數, 會自動建立一個無參數的空白建構函數
      - 宣告建構函數
        - 名稱必需與 class 完全相同
        - 快輸建立方式, 輸入 'ctor' 再按二次 tab
        - 建構函數可以多載 — 限制是參數不可以相同 — 參數數量, 型別順序不能相同
        - 宣告 Member(string name)
        - 宣告 Member(string name, email)
        - 練習叫用上述兩個建構函數
        - 如果有自訂建構函數, 不會自動建立一個無參數的空白建構函數
      - 如何判斷參數要設計在建構函數或 method 裡?
        - 如果資訊是幾乎整個類別必備的, 就設計在建構函數 — 在建立物件時就一併傳入物件裡
        - 如果資訊是某 method 才需要用到的, 就宣告在該 method 的參數 — 叫用 method 才傳入
    - 練習
      - class Invoice 發票類別
        - 屬性
          - int Price — 未稅金額
          - int Tax
            - 營業稅
            - ReadOnly
          - int InclusivePrice
            - 含稅金額
            - ReadOnly
        - 建構函數 — Invoice(int price, double taxRate=0.05)
      - class EStore
        - 在建構函數傳入營業稅率
        - private int CalcTax(int price) — 計算營業稅
        - public Invoicet IssueInvoice(int price) — 開立發票, 傳回 Invoice 型別的物件
  - 提升軟體品質
    - 程式碼好理解
      - why?
        - 容易了解這段程式功能是什麼
        - 明白要改某功能,是不是要改這程式
        - 明白要從哪裡修改,擴展程式
      - how?
        - 加適當的註解
        - 遵守Clean Code原則
          - 命名正確
          - 不要太多層巢狀程式碼
          - 遵守單一職責原則, 函數功能愈單一, 愈好理解,愈容易被reuse
          - ...
    - 程式碼功能正確 — 撰寫單元測試 — 確保程式運行結果如同預期
    - 程式碼行為易於預測
      - 思考method應該回傳什麼型別
      - 誠實函數
  - 單元測試入門
  - 閱讀書籍 Demo01.ReadBooks
  - 擴充方法 Demo02.Extensions
  - 建立類別庫專案
  - Demo03
  - SOLID原則
  - 設計模式
  - Demo05
- ADO.NET 36hrs