Class

Shape
NamedShape
Square
EquilateralTriangular

今日演練程式

- 宣告稱為Shape的類別
- 宣告類別為Shape的7邊形變數
- 宣告稱為NamedShape的類別
- 宣告類別為NamedShape的類別Square
- 宣告類別為Square的正邊形變數test，並呼叫其方法
- ---
- 宣告類別為NameShape的類別EquilateralTriangular
- 宣告類別為EquilateralTriangular的變數triangular，並呼叫其方法

宣告稱為Shape的類別

- 形狀類別- class Shape
  - 變數 numberOfSides
  - 函數 simpleDescription

```
class Shape {
    var numberOfSides = 0
    func simpleDescription() -> String {
        return "A shape with \(numberOfSides) sides."
    }
}
```

宣告類別為Shape的7邊形變數

```
var shape = Shape()
shape.numberOfSides = 7
var shapeDescription = shape.simpleDescription()
```

宣告稱為NamedShape的類別

- 有名稱的形狀- class NamedShape
  - 變數 numberOfSides, name
  - 函數 init
  - 函數 simpleDescription

```
class NamedShape {
    var numberOfSides: Int = 0
    var name: String

    init(name: String) {
        self.name = name
    }

    func simpleDescription() -> String {
        return "A shape with \(numberOfSides) sides."
    }
}
```

類別本身的欄位變數

初始化函數的輸入參數

宣告類別為NamedShape的類別Square

- 正方形- class Square: NamedShape
  - 變數 sideLength
  - init
  - 函數 面積
  - 覆寫函數(override func) simpleDescription

母類別為NamedShape

```
class Square: NamedShape {
    var sideLength: Double

    init(sideLength: Double, name: String) {
        self.sideLength = sideLength
        super.init(name: name)
        numberOfSides = 4
    }

    func area() -> Double {
        return sideLength * sideLength
    }

    override func simpleDescription() -> String {
        return "A square with sides of length
            \(sideLength)."
    }
}
```

母類別中的init函數
設定母類別的變數
name

覆寫母類別中的
simpleDescription函數

宣告類別為Square的正邊形變數test，並呼叫其方法

```
let test = Square(sideLength: 5.2, name: "my test
    square")
test.area()
test.simpleDescription()
```

宣告類別為NamedShape的類別EquilateralTriangular

- 等邊三角形- class EquilateralTriangular: NamedShape
  - 變數 sideLength
  - init
  - 函數 面積
  - 覆寫函數(override func) simpleDescription

```
class EquilateralTriangle: NamedShape {
    var sideLength: Double = 0.0

    init(sideLength: Double, name: String) {
        self.sideLength = sideLength
        super.init(name: name)
        numberOfSides = 3
    }

    var perimeter: Double {
        get {
            return 3.0 * sideLength
        }
        set {
            sideLength = newValue / 3.0
        }
    }

    override func simpleDescription() -> String {
        return "An equilateral triangle with sides
            of length \(sideLength)."
    }
}
```

以周長的新值計算邊長並設定

使用邊長計算周長並設定

宣告類別為EquilateralTriangular
的變數triangular，並呼叫其方法

```
var triangle = EquilateralTriangle(sideLength: 3.1,
    name: "a triangle")
print(triangle.perimeter)
triangle.perimeter = 9.9
print(triangle.sideLength)
```