JAVA 基本觀念-模組化設計

修平科技大學 資訊網路技術系
陳文敬老師 jameschen@hust.edu.tw

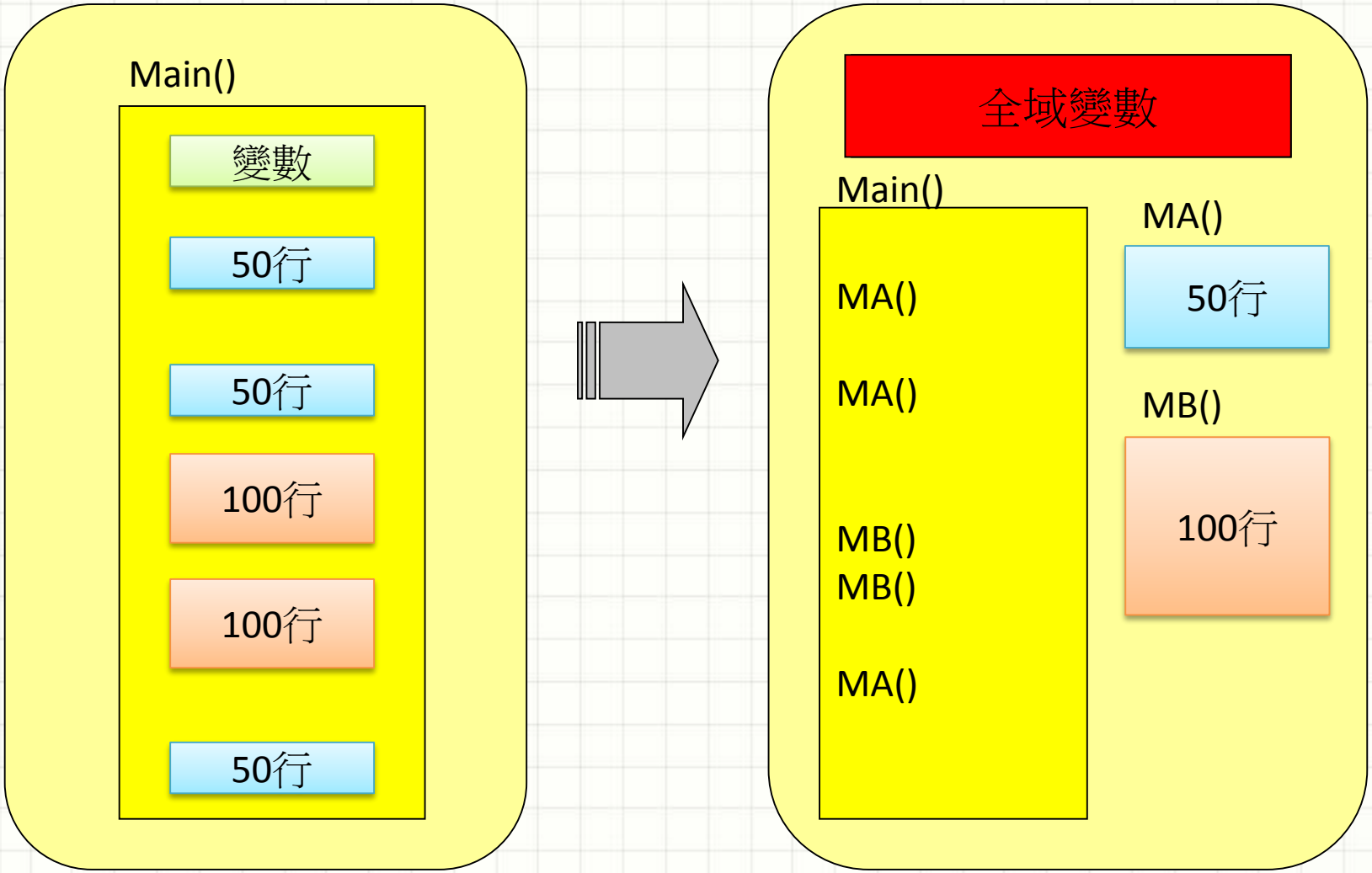
Agenda

1. **模組化設計的好處**
2. **常用的系統內建模組**
   - Math, Wrapper Class (Integer, Float, Double, …)
   - String, StringBuilder, StringBuffer
   - System, Arrays
3. **模組化設計**
   - 自訂方法(Method)、副程式(sub-routine/procedure/function)
   - 自訂類別 (class)、套件(package)
   - 參數的傳遞 : call-by-value, call-by-reference
4. **遞迴的解題技巧**

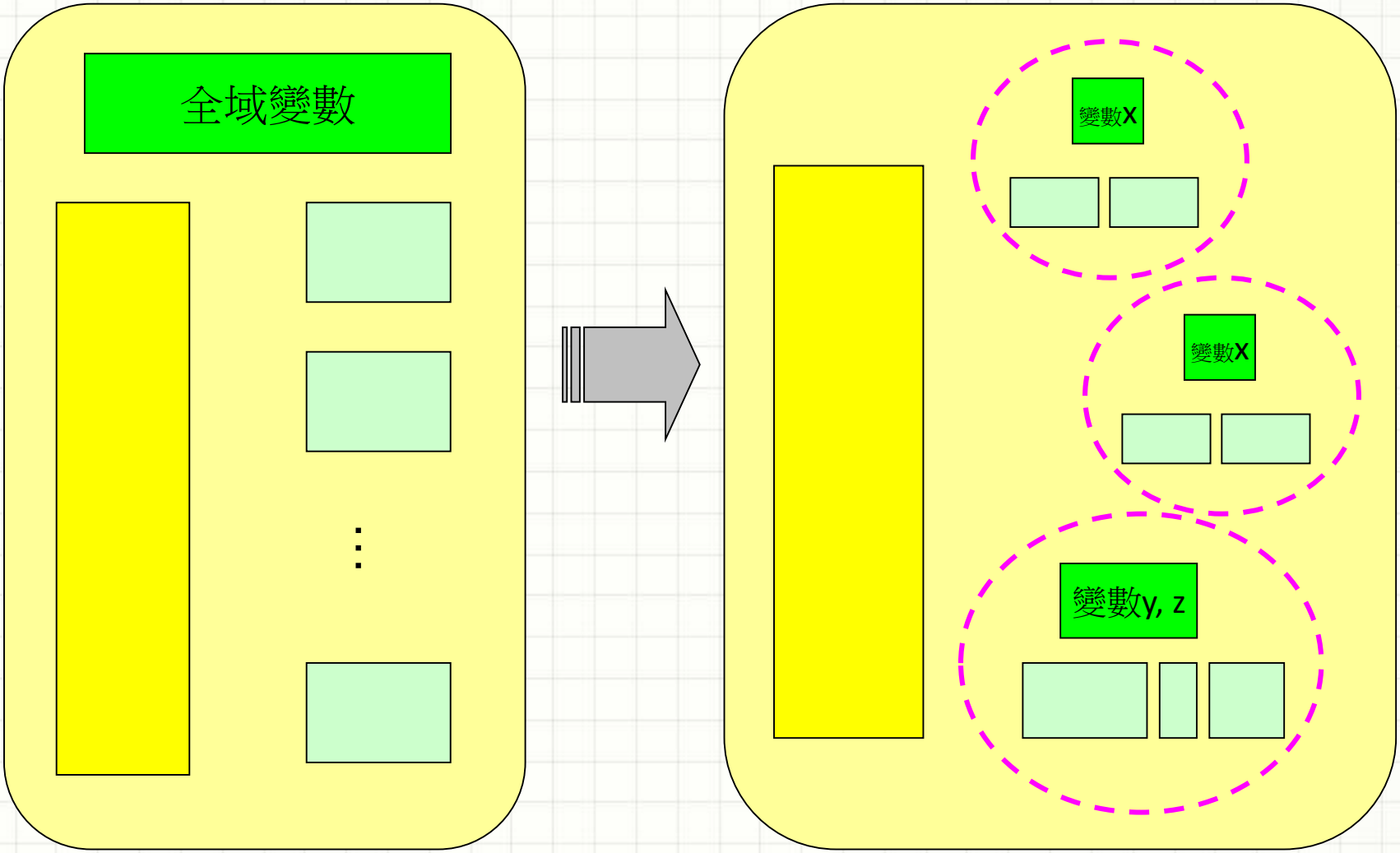
傳統系統開發的常見問題

- **維護**: 不容易修改、維護(牽一髮而動全身)
- **穩定**: 因為系統變大而導致系統不穩定 (人多手雜、誤用資料)、容易崩潰?
- **重用**: 新的專案必須重新開發、重覆使用的機率過低...
- **擴充**: 難以增加系統功能、造成疊床架屋...

程式結構的演變－1 (模組化)



程式結構的演變 -2 (類別/物件化)



學習物件導向技術!!

- **避免不正常的存取：資訊隱藏(Information hiding)**
  - 將相關的資料與處理資料的程式(副程式)加以規範
  - 利用封裝與存取權限加以限制
- **提高程式碼的再利用率 (Reusability)**
  - 元件模組化的設計思維 (component based design)
  - 以類別(物件)為最基本模組，包含相關方法(副程式)
  - 利用繼承機制來改良舊有功能或擴充新功能。
- **利用抽象化來提高系統的可擴充性 (Extensibility)**
  - 利用標準介面來降低元件之間的藕合度 (loosely couple)

先學好模組化設計 … 好處是:

1. 能夠重複使用、分享(共用)
2. 方便維護(修改)、確保程式一致性
3. 程式碼變簡潔(變小)
4. 方便閱讀程式
5. … …

常用的內建模組

- 數學運算 : Math
- 外覆型別 : Integer, Float, Double, …
- 字串處理 : String
- 日期 : Date, Calendar

Math 數學模組

Constant	Description
`Math.E`	2.7182818...
`Math.PI`	3.1415926...

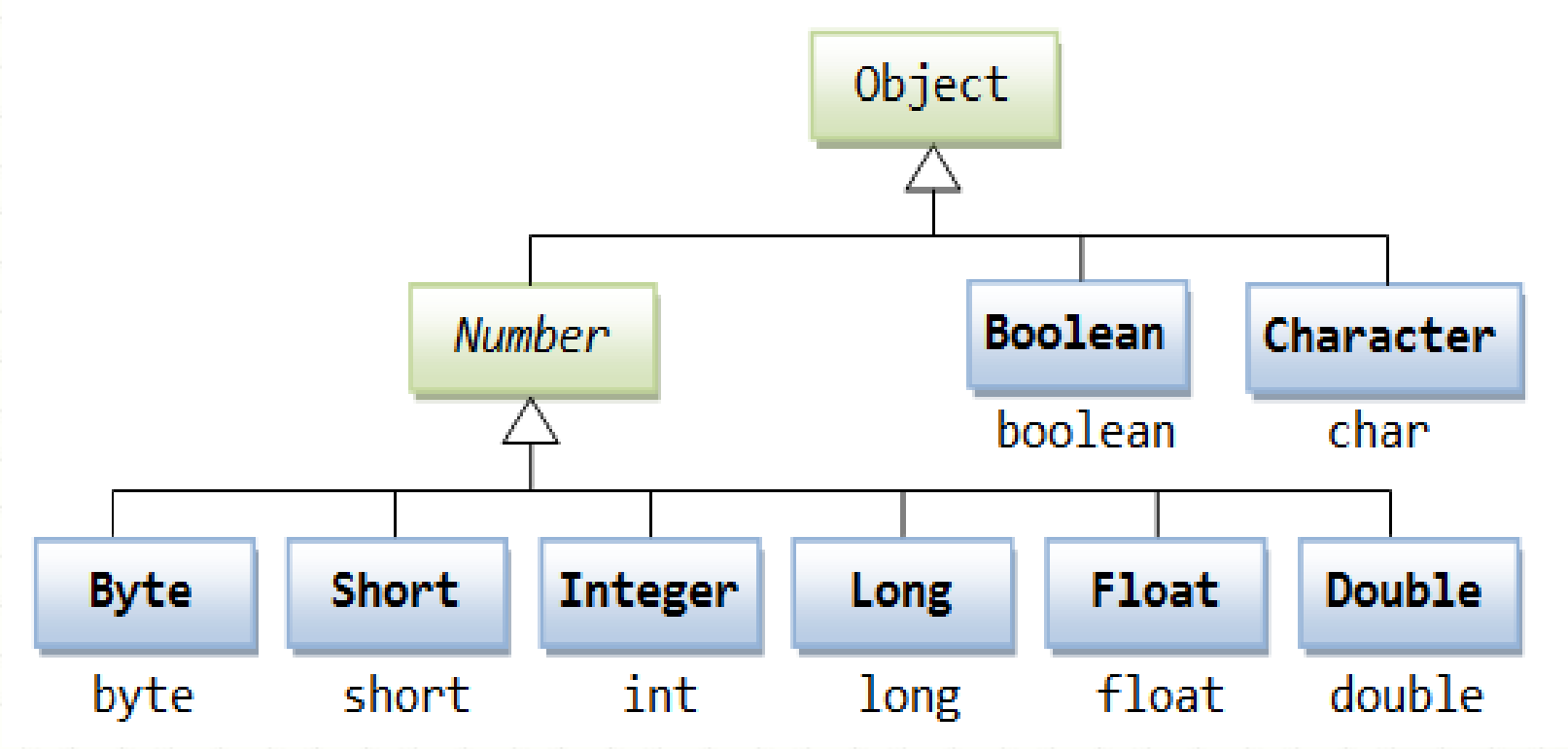
回傳值	方法名稱	說明
`double`	`Math.abs(`*value*`)`	absolute value
`double`	`Math.ceil(`*value*`)`	rounds up
`double`	`Math.floor(`*value*`)`	rounds down
`double`	`Math.log10(`*value*`)`	logarithm, base 10
`double`	`Math.max(`*value1*`, `*value2*`)`	larger of two values
`double`	`Math.min(`*value1*`, `*value2*`)`	smaller of two values
`double`	`Math.pow(`*base*`, `*exp*`)`	*base* to the *exp* power
`double`	`Math.random()`	random `double` between 0 and 1
`double`	`Math.round(`*value*`)`	nearest whole number
`double`	`Math.sqrt(`*value*`)`	square root

`double`
`double`
`double`
	`Math.sin(`*value*`)`
	`Math.cos(`*value*`)`
	`Math.tan(`*value*`)`
		sine/cosine/tangent of an angle in radians

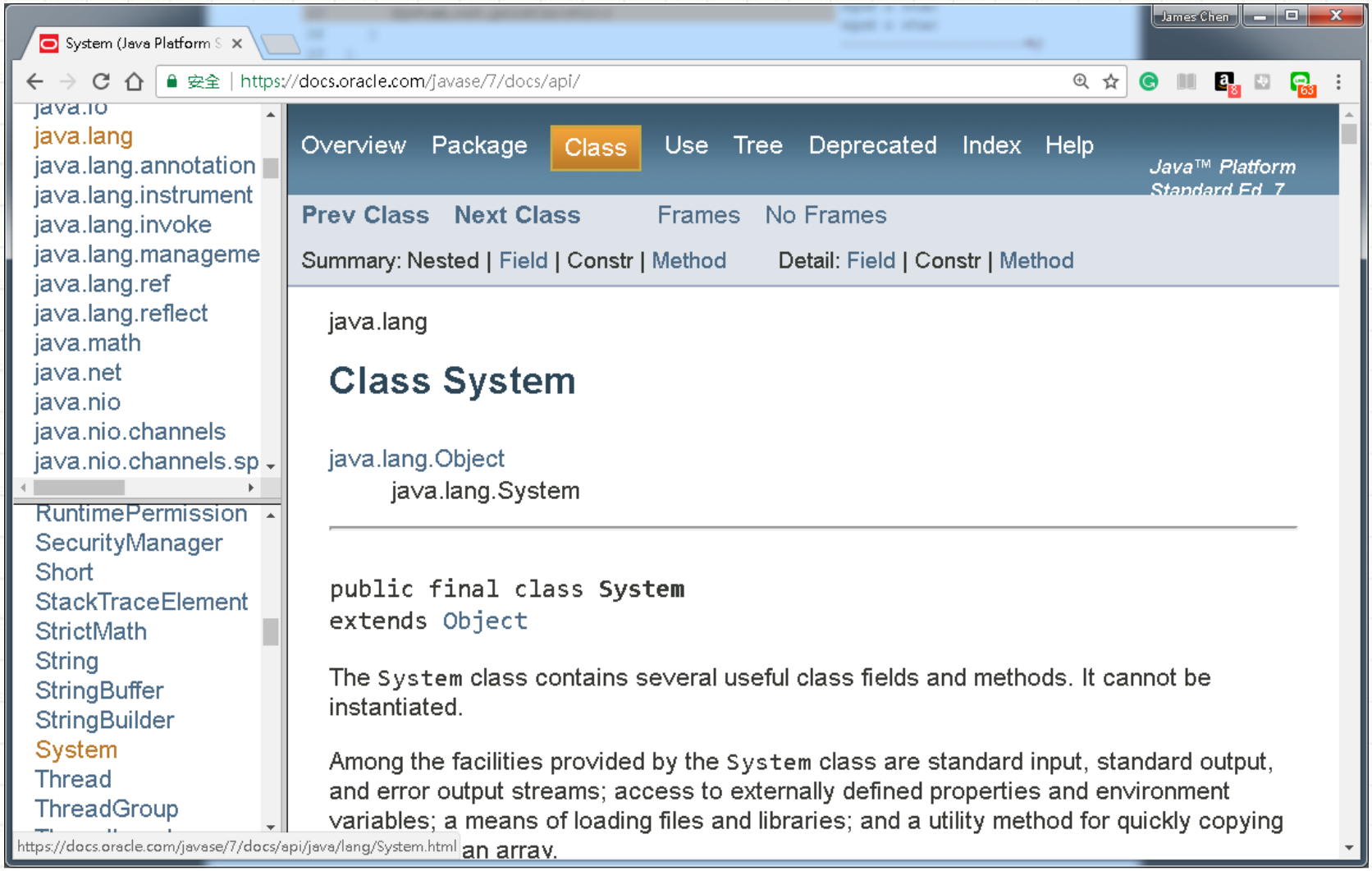
`double` `double`	`Math.toDegrees(`*value*`)` `Math.toRadians(`*value*`)`	convert degrees to radians and back

外覆型別(Wrapper Classes)

- 資料轉換與解析

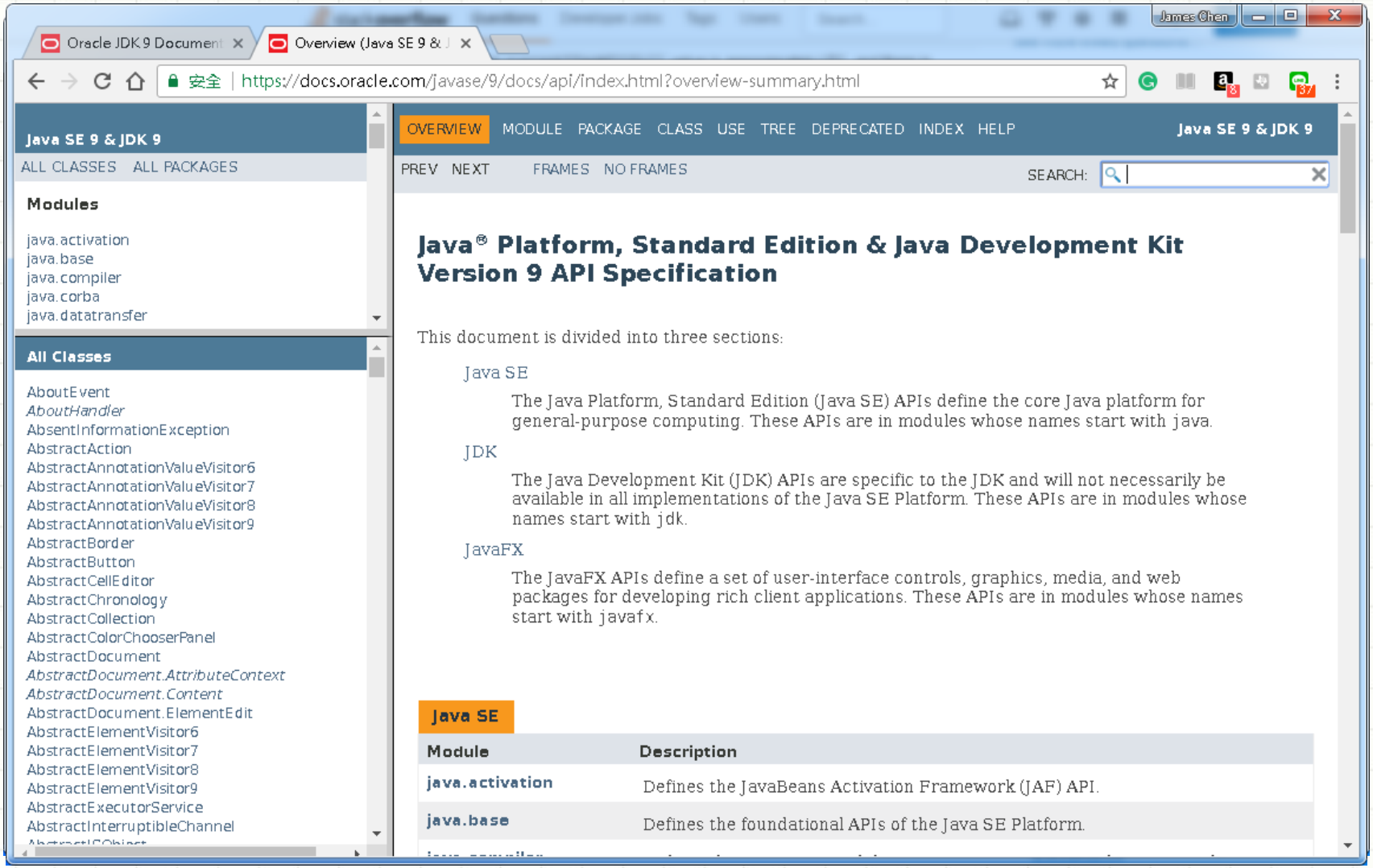


學習查詢 Java Docs (APIs)



https://docs.oracle.com/javase/7/docs/api/

學習查詢 Java Docs (APIs)



https://docs.oracle.com/javase/9/

練習時間：樂透開獎

- 利用系統模組寫一個**樂透開獎**
- 要求：由 1~49之間選取六個相異的數字
- 作法：利用系統內部模組 ***Math.random()***

(int) (Math.random() * 49 + 1)

自訂模組(方法, Method) – 不帶參數

- 決定方法的名稱、執行參數、回傳值

```
public static void printTriangle( ) {
        for(int i=1; i<=7 ; i++) {
            for(int j=1; j<=i; j++) {
                System.out.print("*");
            }
            System.out.print("\n");
        }
        System.out.print("\n");
    }
```

```
*
**
***
****
*****
******
*******
```

自訂模組(方法, Method) –有參數

```
1.  public static void printTriangle(int n) {
2.          for(int i=1; i<=n ; i++) {
3.              for(int j=1; j<=i; j++) {
4.                  System.out.print("*");
5.              }
6.              System.out.print("\n");
7.          }
8.          System.out.print("\n");
9.  }
```

```
printTriangle( 3 ); printTriangle( 5 );
```

```
*
**
***
*
**
***
****
*****
```

自訂工具類別(Class)：MyMath

1. 將相關、有用的方法集合在一起(封裝在一個類別之內)，方便未來可以重複使用
2. 必須注意**參數的傳遞**與**資料回傳**的方式！

```
public class MyMath {
    public static int factorial(int x) { /*階乘*/ }
    public static boolean isPrime(int x) { /*質數*/ }
    public static boolean isLeapYear(int year) {/*閏年*/}
    public static int[] getPrimeFactor(int n) {/*質因數*/}
    public static int search ( int[] data, int x) { /*一般搜尋 */ }
    public static void sort (int[] data) {  /* 排序 */ }
    public static void binarySearch (int[] data, int x) {  /* 二元搜尋 */ }
}
```

參數: Call by value ?
Call by reference?

練習時間：開發工具類別(*MyMath*)

- 下載練習專案 Ex02060X.zip
- **匯入專案** : [File] -> [import…] -> [General] -> [Existing project …] -> [Select archive file…] : 選擇剛剛下載的zip
- 建立 ***MyMath*** 的說明文件(Javadoc → HTML)
  - 查看 ***MyMath*** 類別內的 JavaDoc 寫法
  - 生成 JavaDoc : [Project] -> [Generate Javadoc…]
- **執行 Main.java 看看** :
  - 測試已經完成的方法 *isPrime(…), isLeapYear(…)*
- **嘗試編寫其他方法**
  - ***factorial****(…) : 階乘*
  - ***search****(…) :搜尋資料*
  - ***sort****(…) : 排序(由小排到大)*
  - ***binarySearch****(…) : 另類搜尋方法(二元搜尋)*

階乘 *factorial*

- 階乘定義

  5! = 1*2*3*4*5

  N! = 1*2*3*4*…*N

- 完成factorial 方法設計 :利用 ***return*** 回傳結果

```
public static int factorial(int n) {
        int result = 1;
        for (int i=1; i<= n ; i++) {
            result = result * i;
        }
        //
        return result ;
}
```

搜尋資料 search

- 從一堆資料(陣列 data)中尋找指定的數字，若找到 x, 則回傳x 的註標，否則回傳-1

```
public static int search(int[] data, int x) {
    for (int i=0; i<data.length; i++) {
        if (data[i] == x ) {  // 找到 x !
            return i;
        }
    }
    // 找不到 x
    return -1;
}
```

排序(Sort): 將資料由小排到大

- **排序 (Sorting) : 處理數值資料時常常使用**
  - 將一堆數字由小到大依序排列!

原本： 15, 30, 7, 43, 21, 1

最終： 1, 7, 15, 21, 30, 43

縮小問題：先找出第1小的數字!

- 原本： 15, 30, 7, 43, 21, 1
- 最終： 1,

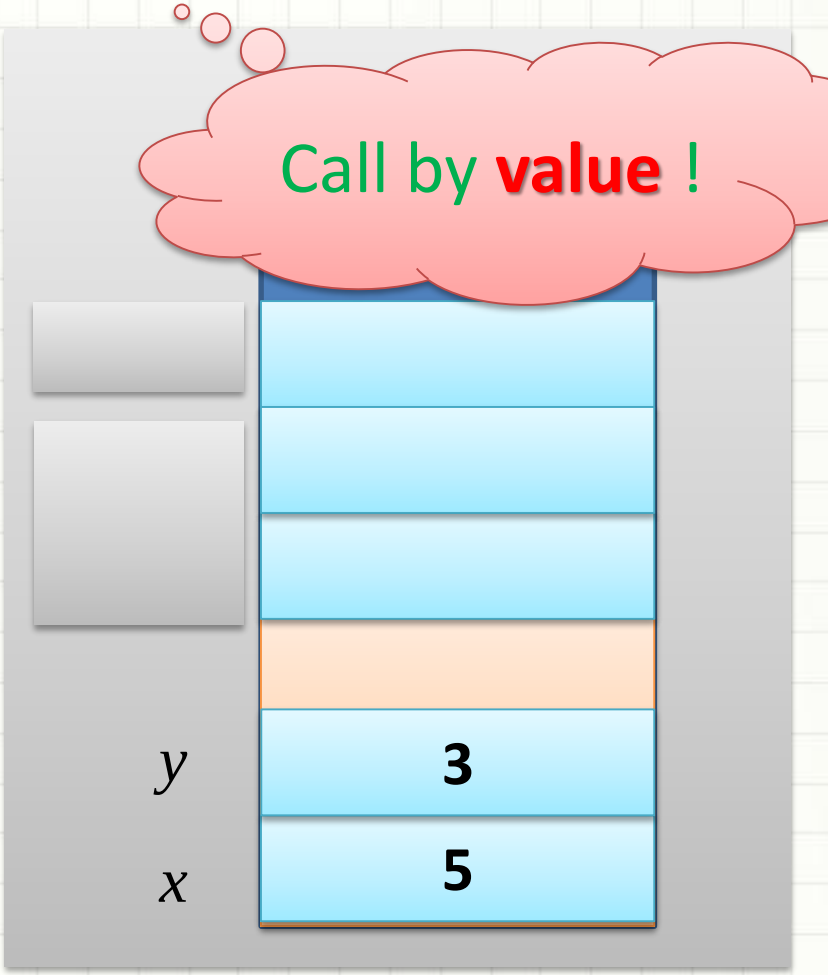
選擇排序 sort (需要改寫)

```
public static void sort(int[] data) {
    // 找出最小的數，放到 位置 0 (i)
       int k = 0;
       for (int j=k+1; j<data.length; j++) {
           if (data[j] < data[k] ) {  // 有更小的數?
             // 交換  k, j 位置的內容



           }
       }

}
```

思考一個小問題：a、b內容互換

```
1.  public static void swap (int a, int b) {
2.          int temp;   // 暫存
3.          temp = a;
4.          a = b;
5.          b = temp;
6.  }
```

```
// 呼叫 swap( …) ?
int x=5, y =3;
swap( x, y );
```



傳遞陣列參數(變動資料大小)?

```
1.  public static int search (int[ ] data, int x) {
2.        // 從 陣列data 找出 x
3.        // 回傳 x 的位置
4.        // 若找不到 x, 就回傳 -1
5.        // 關鍵字: return
6.  }
```

Array:
Call by **reference**!

```
// 呼叫 search( … ) ?
int[ ] list = { 2, 32, 9, 38, 39, 13, 25, 50};
int position = search( list, 13);
```

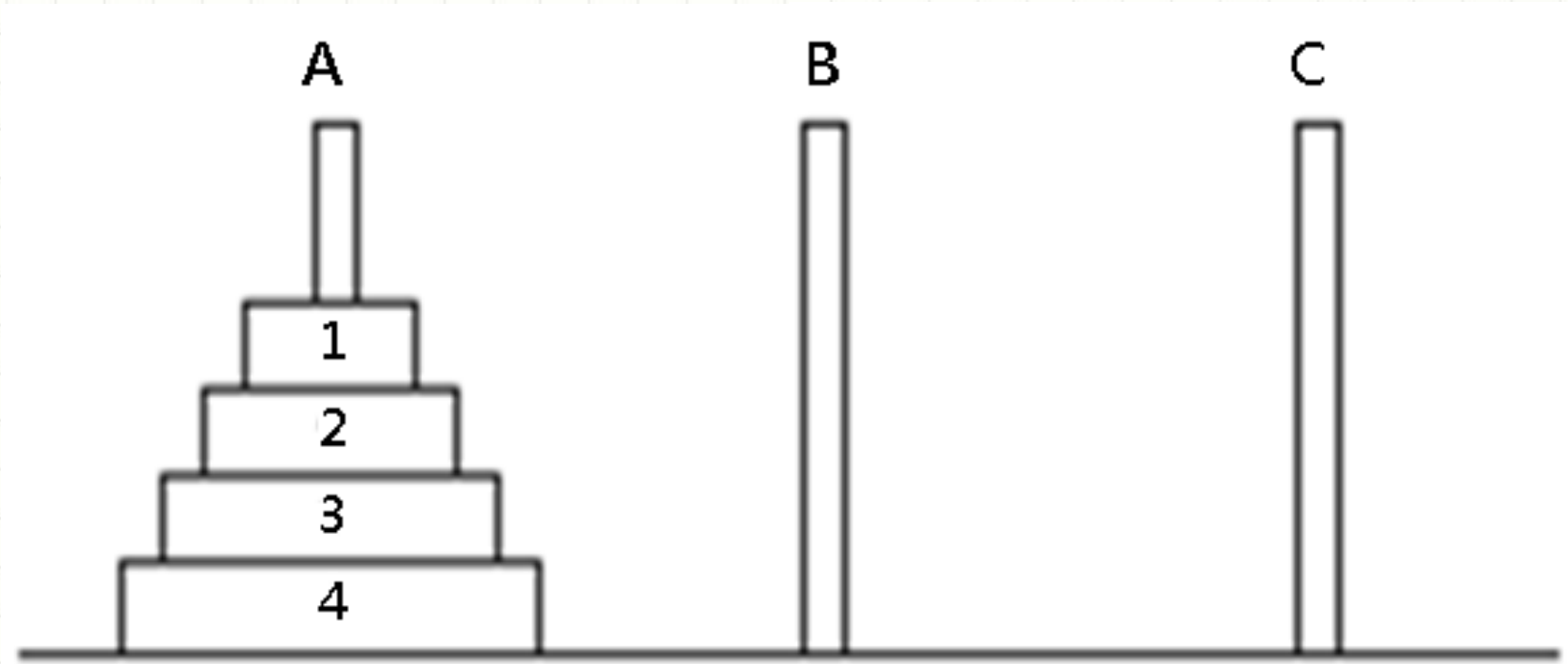
了解 Java 的程式進入點 main()

```
public class Main {
    public static void main(String[] args) {
        for (int i=0; i<args.length; i++) {
            System.out.println(i+“個參數:"+ args[i]);
        }
    }
}
```

執行看看: *java Main 123 456 ABC "XYZ 789"*

遞迴的迷人之處

- **遞迴**：將大問題拆為小問題、先解出小問題，就能湊出大問題的答案；而問題夠小就一定有答案。



利用遞迴解河內之塔問題

```
public static void hanoi(int n, char from , char to , char temp ) {
    if (n == 1) {  // (A)小問題:直接就有答案
        //將盤子n (=1)從柱子(from)搬到 (to) ;
    }
    else {          // (B)大問題: 利用遞迴先解較小的問題
        hanoi(n-1, from, temp, to);   // 先暫時搬走上面的 n-1個盤子
        //將盤子n (=?)從柱子(from)搬到 (to) ;
        hanoi( n-1, temp, to, from);   // 再搬回上面的 n-1個盤子
    }
}
```

利用**遞迴**改寫**階乘**的計算?!

- 再看一次階乘的定義

  1! = 1

  2! = 1 * 2 = 1! * 1

  3! = 1 * 2 * 3 = 2! * 3

  4! = 1 * 2 * 3 * 4 = 3! * 4

  N! = 1*2*3*4*…*N = (N-1)! * N

factorial(n) = 1 , if n =1

factorial(n) = factorial(n-1) * n, if n >=2

階乘 *factorial* :遞迴版本

- factorial(n) = 1 , if n =1
- factorial(n) = factorial(n-1) * n, if n >=2

```
public static int factorial(int n) {
    if (n == 0 || n == 1)
        return 1;
    else if (n >= 2)
        return factorial(n-1) * n;
}
```

遞迴：
自己呼叫自己

練習時間： *power(x, n)*

- 迴圈(迭代)的版本

```
public static int power(int x, int n) {
    int result = 1;
    for (int i=0; i<n; i++) {
        result *= x;
    }
    return result;
}
```

練習時間：改寫 *power(x, n) - 1*

- 遞迴的版本1

```
public static int power(int x, int n) {



}
```

練習時間：改寫 *power(x, n)* - 2

- 遞迴的版本2 (效率比較好)

```
public static int power(int x, int n) {



}
```