



# PROJECT 4

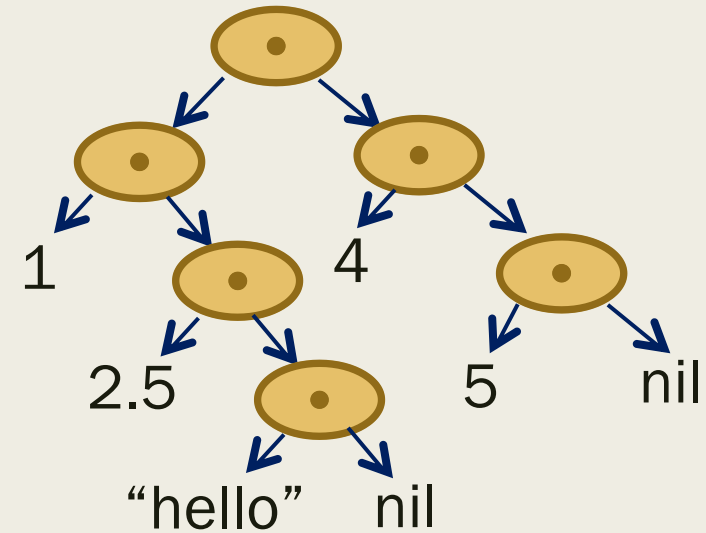
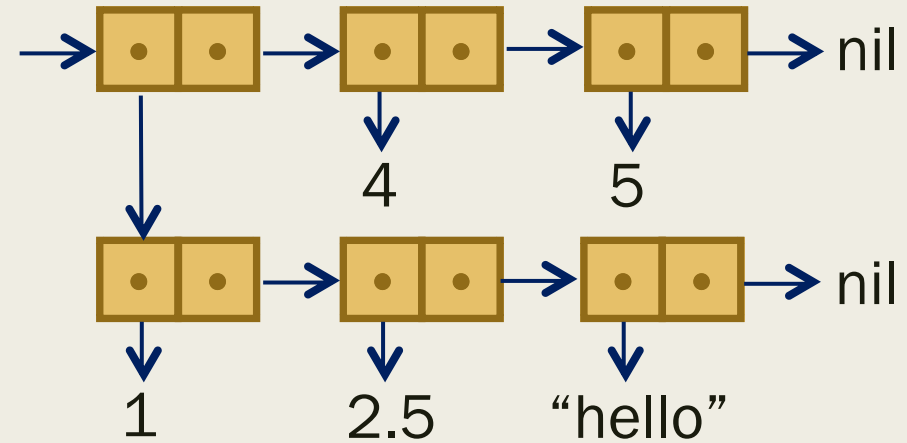


# Project 4

- 在Project 3的基礎上，實作十個系統函數: **read, write, eval, set!, create-error-object, error-object?, display-string, newline, symbol->string, number->string**

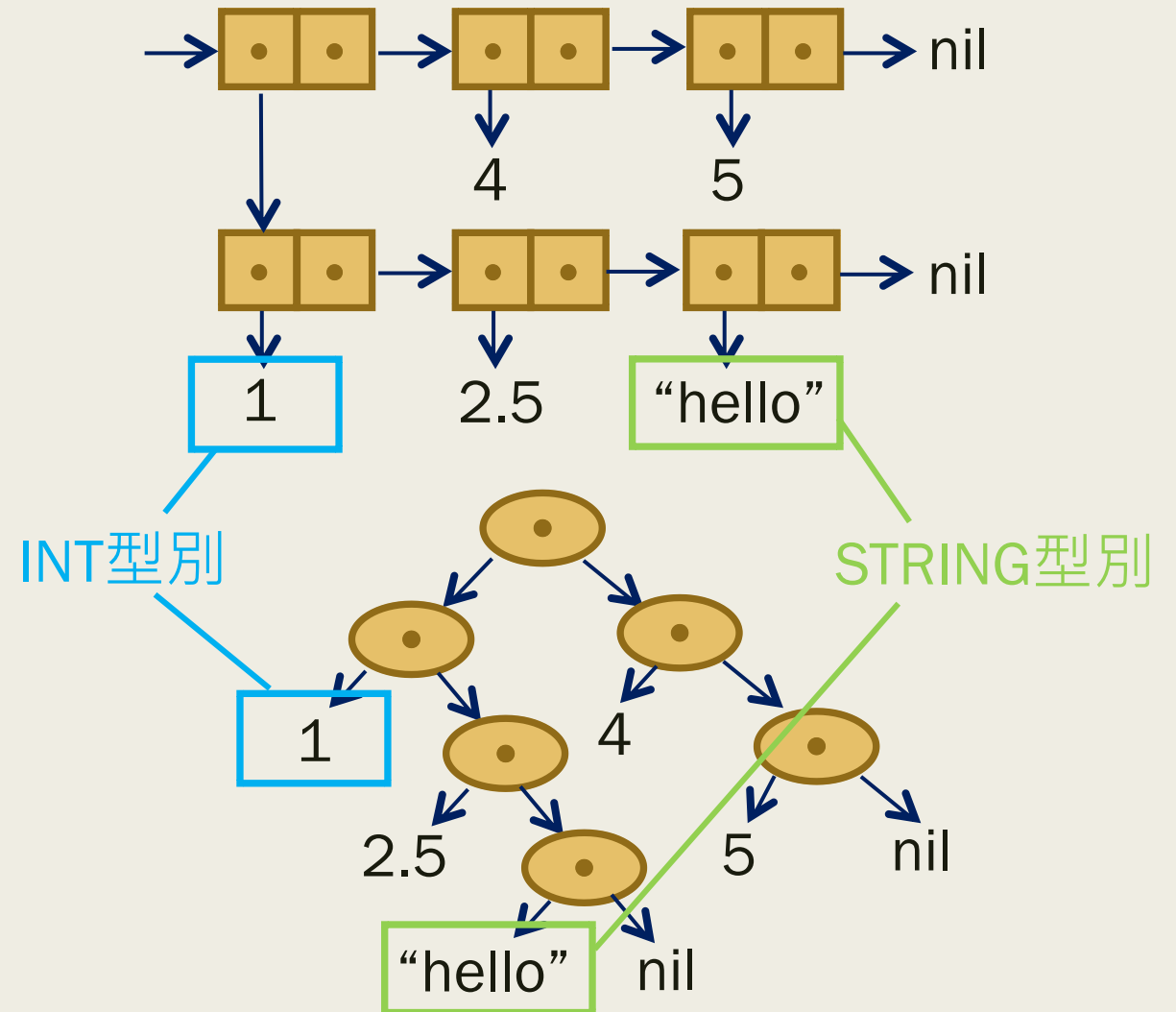
- create-error-object
- error-object?

```
(( 1 2.5 "hello" ) . ( 4 . ( 5 . nil ) ) )
```



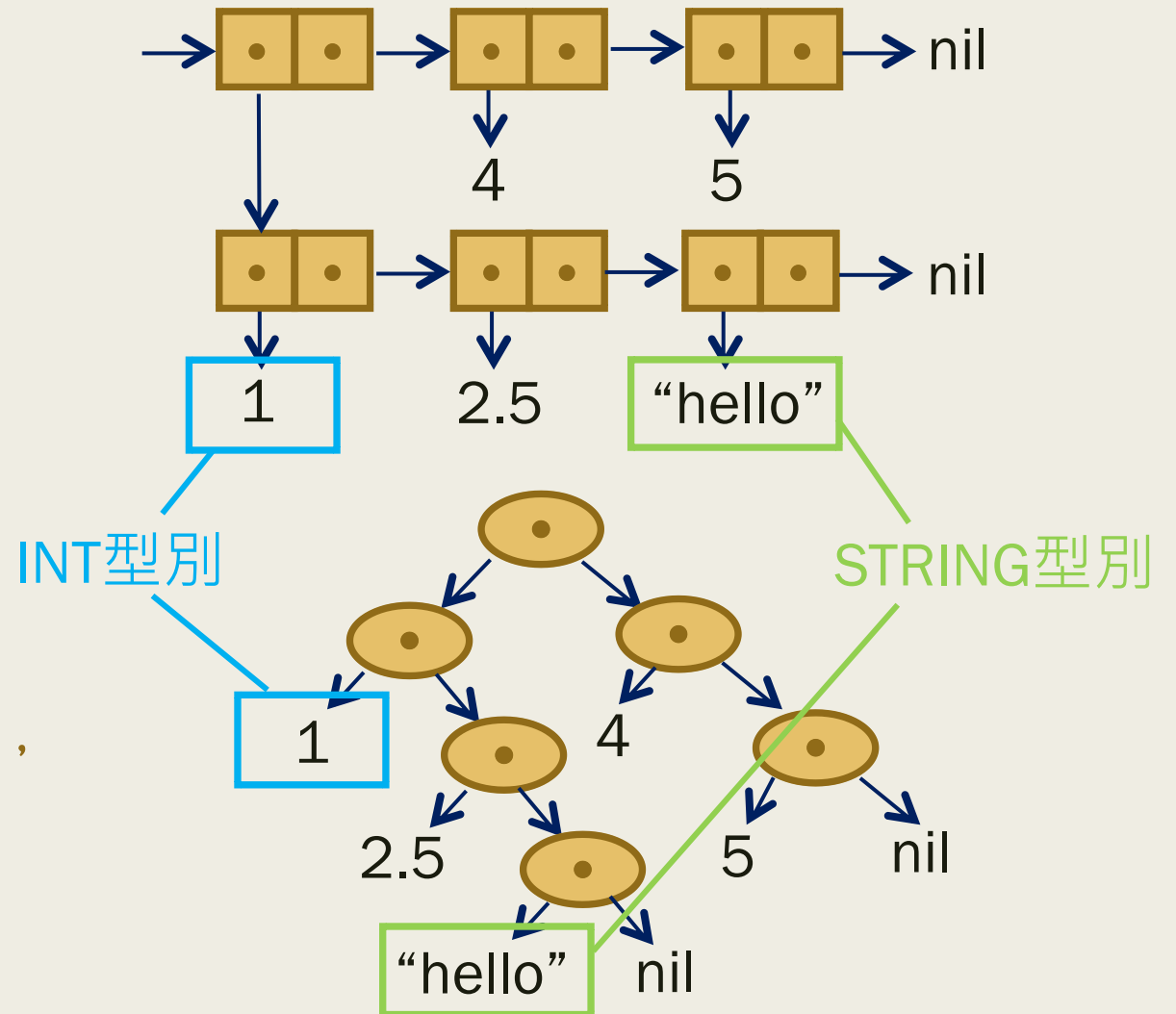
- create-error-object
- error-object?

(( 1 2.5 "hello" ) . ( 4 . ( 5 . nil ) ) )



- create-error-object
- error-object?

(( 1 2.5 "hello" ) . ( 4 . ( 5 . nil ) ) )



新增一個type，名為ERROR，  
存的值與STRING相同

## ■ create-error-object

## ■ error-object?

```
> (create-error-object "hello")
```

```
"hello"
```

```
> (define a (create-error-object "abc"))
```

```
a defined
```

```
> a
```

```
"abc"
```

```
> (error-object? a )
```

```
#t
```

ERROR型別的object與STRING型別的object有幾乎一樣的行為，唯二不同：

1. ERROR型別的object在error-object?函式的檢測下結果為#t，STRING則為nil
2. STRING型別可被讀入的方式生成，而ERROR型別只能呼叫create-error-object生成、或使用read函數時發生錯誤來生成

## ■ read

是OurScheme的主要input函數，功能為嘗試從input讀取下個S-expression，並以讀到的S-expression作為回傳值。

read函數會檢查輸入S-expression文法，若檢查到錯誤，將生成值為錯誤訊息的error-object回傳

read函數不接受任何參數

## ■ read

```
> (define a (read)) (1 2 3)
```

```
a defined
```

```
> a
```

```
( 1
```

```
  2
```

```
  3
```

```
)
```

```
> (error-object? a)
```

```
nil
```

```
> (define a (read))( 1 3 5 . 7 s )
```

```
a defined
```

```
> a
```

```
"ERROR (unexpected character) : line 1 column 13 character 's'"
```

```
> (error-object? a)
```

```
#t
```



## ■ read

```
> (define a (read)) (1 2 3)
```

```
a defined
```

```
> a
```

```
( 1
```

```
  2
```

```
  3
```

```
)
```

```
> (error-object? a)
```

```
nil
```

```
> (define a (read))( 1 3 5 . 7 s )
```

```
a defined
```

```
> a
```

```
"ERROR (unexpected character) : line 1 column 13 character 's'"
```

```
> (error-object? a)
```

```
#t
```

- write
- display-string
- newline

是OurScheme的主要的三個output函數，功能皆為print內容

## ■ write

write接受一個參數，該參數為一個S-expression，功能為print出這個S-expression，其函數回傳值為其給定的S-expression參數

```
> (write '(1 2 3))
```

```
( 1
```

```
  2
```

```
  3
```

```
)( 1
```

```
  2
```

```
  3
```

```
)
```

```
> (write "hi")
```

```
"hi""hi"
```

## ■ write

write接受一個參數，該參數為一個S-expression，功能為print出這個S-expression，其函數回傳值為其給定的S-expression參數

```
> (write '(1 2 3))
```

```
( 1
```

```
 2
```

```
 3
```

```
)( 1
```

```
 2
```

```
 3
```

```
)
```

```
> (write "hi")
```

```
"hi" "hi"
```

## ■ display-string

display-string同樣接受一個參數，但該參數只能是string-object或是error-object，功能為print出其不帶引號的字串值，函數回傳值為給定的參數本身

```
> (display-string "hi")
```

```
hi"hi"
```

```
> (display-string (write "hi"))
```

```
"hi"hi"hi"
```

**write vs. display-string**

```
> (write "hello")
```

```
"hello""hello"
```

```
> (display-string "hello")
```

```
hello"hello"
```

## ■ display-string

display-string同樣接受一個參數，但該參數只能是string-object或是error-object，功能為print出其不帶引號的字串值，函數回傳值為給定的參數本身

```
> (display-string "hi")
```

```
hi"hi"
```

```
> (display-string (write "hi"))
```

```
"hi"hi"hi"
```

### write vs. display-string

```
> (write "hello")
```

```
"hello""hello"
```

```
> (display-string "hello")
```

```
hello"hello"
```

## ■ newline

newline不接受任何參數，其功能為print出一個line-enter字元至output，函數回傳值為nil

```
> (newline)
```

```
nil
```

```
> (begin (write "hi") (newline) (display-string "hi") (newline))
```

```
"hi"
```

```
hi
```

```
nil
```

以上output函數輸出的字串內容: "hi"**\n**hi**\n**

- symbol->string
- number->string

接受一個symbol型別或是number型別參數，將參數轉成字串，symbol->string

只接受symbol型別的參數、number->string只接受number型別的參數

```
> (symbol->string 'Hi)
```

```
"Hi"
```

```
> (number->string 12345)
```

```
"12345"
```

```
> (symbol->string ( quote sym ) )
```

```
"sym"
```

```
> (number->string 25.07)
```

```
"25.070"
```



## ■ eval

eval接受一個參數，參數為一個S-expression，主要功能為對這個參數做evaluate，同你在OurScheme輸入完一段S-expression後系統做的事，該函數回傳值為evaluate後的結果。

> '(car '(1 2 3))	> (eval '(car '(1 2 3)))
( car	1
( quote	
( 1	
2	> (car '(1 2 3))
3	1
)	
)	
)	

## ■ set!

set!是個functional forms，會檢查format。

接受兩個參數，第一個參數必須是個symbol，第二個參數是任一個S-expression，其函數功能為evaluate第二個參數後，將evaluate結果值設為第一個參數symbol的binding，並將這個evaluate結果值當作此函數的回傳值。

set!基本上是另一個版本的define，不同之處在於set!有回傳值、且可在各level層使用(define只能在top level)。

## ■ set!

```
> (set! a '(1 2 3))
```

```
( 1
```

```
  2
```

```
  3
```

```
)
```

```
> a
```

```
( 1
```

```
  2
```

```
  3
```

```
)
```

```
> (cons (set! a '(4 5))
```

```
      a
```

```
    )
```

```
  (( 4
```

```
    5
```

```
  )
```

```
4
```

```
5
```

```
)
```

在此定義a的binding為(4 5)，以(4 5)  
作為(set! a '(4 5))的回傳值

在此使用a，evaluate過後結果為(4 5)

## ■ set!

set!會優先修改區域變數，若不存在區域變數，才定義全域變數

```
> (define (fun a) (set! a 90) (set! b 100))
```

```
fun defined
```

```
> a
```

```
ERROR (unbound symbol) : a
```

```
> b
```

```
ERROR (unbound symbol) : b
```

```
> (fun 10)
```

```
100
```

```
> a
```


```
ERROR (unbound symbol) : a
```

```
> b
```

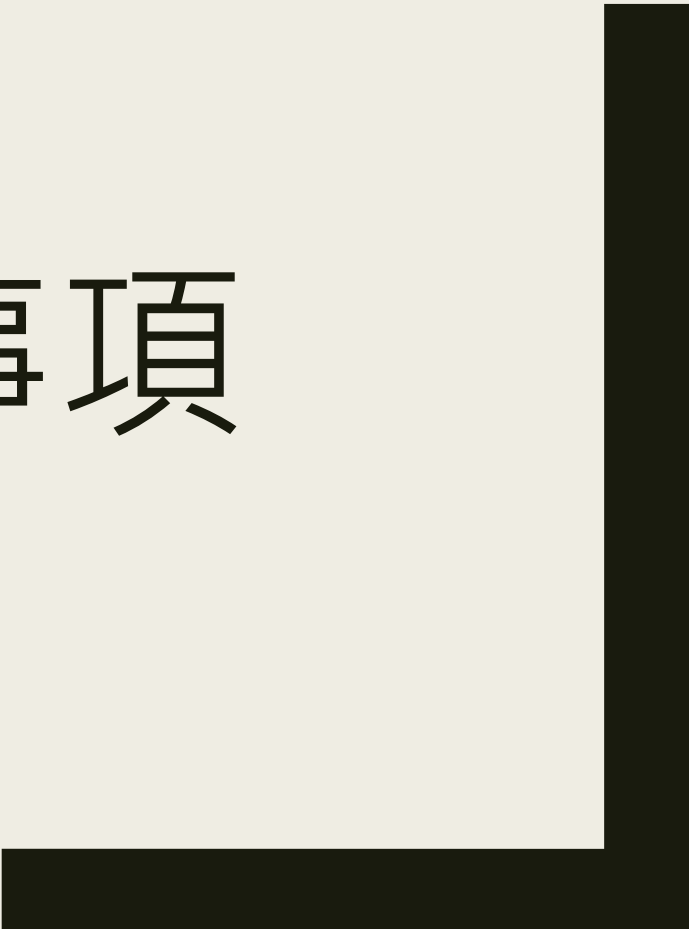
```
100
```

# 擴展的指令

- create-error-object
- error-object?
- read
- write
- display-string
- newline
- symbol->string
- number->string
- eval
- set!



# PL系統注意事項



# C++經驗法則

- 如何排版
  - 使用排版工具，可將C++程式排版成符合PL要求的格式，包括縮排和註解  
<https://github.com/jason89923/CRACKCAL.git>
- 不穩定出現的safe code
  - 可以申請增加程式runtime
- 出現safe code但不知道行數
  - 一般發生在遞迴函數中

# C++ 經驗法則-如何避免safe code

- 甚麼是safe code?
- Safe code **vs** segmentation fault?



# C++ 經驗法則-如何避免safe code

- 不要用map
  - `map<string, string>`
- Vector要用new的，尤其做為data member時
  - `vector<string> v; // 錯誤`
  - `vector<string>* v_ptr = new vector<string>* (); // 正確`
- 不要深度操作陣列
  - `int array[100]; &array[10];`

# C++ 經驗法則-如何避免safe code

## ■ 不要深度操作指標

- `int * intPtr = new int(10); *intPtr`
- `int a = 10; &a`
- 函數指標 `typedef int (*operation)(int, int);`
- 萬用指標 `void*`
- 多型父類別轉子類別
- 以上寫法都是不能用的

# java經驗法則-字串處理

- 數字、字串轉換方便，且字串處理容易
  - `int n1 = Integer.parseInt( "256" );` *// string to int*
  - `int n2 = Integer.parseInt( "-5" );`
  - `int n3 = Integer.valueOf( "0" );`
  - `String s = String.valueOf( -100 );` *// int to string*
- 注意事項， 以下寫法會產生`java.lang.NumberFormatException`
  - `Integer.parseInt( "+256" );`
  - `Integer.parseInt( "+1" );`
  - `Integer.parseInt( "+0" );`
  - 事先拿掉字串前 '+' 號字元即可

# java經驗法則-Regular expression

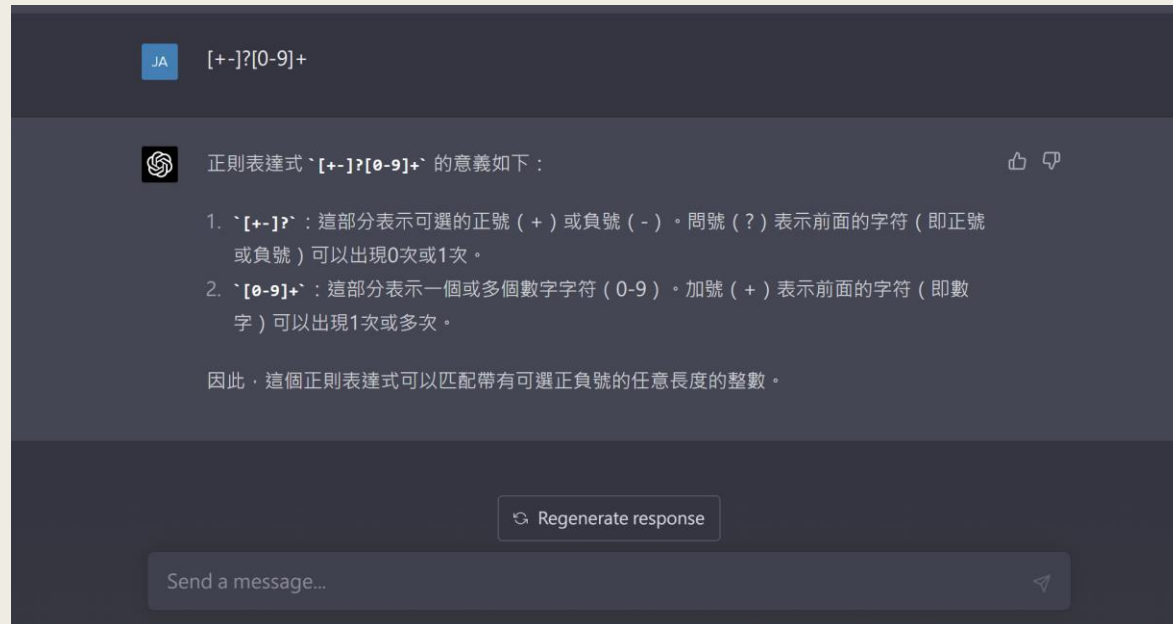
- Regular expression
  - 數字: `"\d"`
  - 任意長度的數字: `"\d+"`
  - 包含+或-: `"[+-]"`
  - 限制+或-只能出現一次: `"[+-]?"`
  - 英文: `"[a-zA-Z]"`
  - 任意長度的英文: `"[a-zA-Z]+"`
  - 多問問chatGPT
- 利用Regular expression檢查數字

```
String s1 = "123457" ;  
String s2 = "+0.5" ;  
System.out.println( s1.matches( "[+-]?[0-9]+" ) ) ;  
System.out.println( s2.matches( "[+-]?[0-9]+" ) ) ;
```

# java經驗法則-Regular expression

- Regular expression
  - 數字: `"\d"`
  - 任意長度的數字: `"\d+"`
  - 包含+或-: `"[+-]"`
  - 限制+或-只能出現一次: `"[+-]?"`
  - 英文: `"[a-zA-Z]"`
  - 任意長度的英文: `"[a-zA-Z]+"`
  - 多問問chatGPT
- 利用Regular expression檢查數字

```
String s1 = "123457";  
String s2 = "+0.5";  
System.out.println( s1.matches( "[+-]?[0-9]+" ) );  
System.out.println( s2.matches( "[+-]?[0-9]+" ) );
```



# java經驗法則-PAL不提供Enum

- PAL系統不提供enum功能，可以用以下的方法取代

```
class G {  
    public static final int RED = 1 ;  
    public static final int BLUE = 2 ;  
    ...  
} // class G
```

# java經驗法則-PAL開放且常用類別

- PAL開放使用的常用類別

- *Vector*
- *ArrayList*
- *Stack*
- *LinkedList*

- 使用Stack時，需特別注意

- 不開放諸如 `isEmpty()`, `clear()`, `add()` ... 等其父類別實作的方法
- *PAL內部只能使用以下方法*

1. `boolean empty()`
2. `Object peek()`
3. `Object pop()`
4. `Object push(Object element)`
5. `int search(Object element)`

# java經驗法則-Call by Reference技巧

- 利用CYICE函式庫BooleanObj類別

```
import CYICE.BooleanObj;

class A {
    public void Foo1() throws Throwable {
        BooleanObj b = new BooleanObj( false ) ;
        System.out.println( b.val ) ;
        ChangeBoolean( b ) ;
        System.out.println( b.val ) ;
    } // Foo1()

    public void ChangeBoolean( BooleanObj b ) throws Throwable {
        b.val = ! b.val ;
    } // ChangeBoolean()
} // class A
```



# java經驗法則-CYICE函式庫



# java經驗法則-CYICE函式庫用法

## ■ Import CYICE的ICEInputStream類別

```
import CYICE.ICEInputStream ;

class G {
    public static ICEInputStream sIn ;
    public static int sTestNum ;

    public static void Init() throws Throwable {
        sIn = new ICEInputStream() ;
    } // Init()
} // class G
```

# java經驗法則-CYICE函式庫文檔

## ■ 點選index.html

名稱	大小	封裝後	類型	修改的日期	CRC32
..			檔案資料夾		
resources	57	57	檔案資料夾	2023/3/25 下午...	
index-files	158,006	27,747	檔案資料夾	2023/3/25 下午...	
CYICE	483,179	78,033	檔案資料夾	2023/3/25 下午...	
stylesheet.css	1,420	433	階層式樣式表文件	2008/6/4 上午 ...	63F1F4C0
serialized-form....	6,883	1,442	Chrome HTML Do...	2008/6/4 上午 ...	57BA8719
package-list	7	9	檔案	2008/6/4 上午 ...	3C66FC36
overview-tree ht...	7,920	1,673	Chrome HTML Do...	2008/6/4 上午 ...	A9D43130
index.html	1,233	640	Chrome HTML Do...	2008/6/4 上午 ...	B1FD0661
help-doc.html	9,485	2,701	Chrome HTML Do...	2008/6/4 上午 ...	8F9181E1
deprecated-list....	5,053	1,167	Chrome HTML Do...	2008/6/4 上午 ...	9D95A087
constant-values....	5,117	1,165	Chrome HTML Do...	2008/6/4 上午 ...	D2CEF48B
allclasses-nofra...	2,024	648	Chrome HTML Do...	2008/6/4 上午 ...	7E597F20
allclasses-frame...	2,384	664	Chrome HTML Do...	2008/6/4 上午 ...	CABB79D3

# java經驗法則-CYICE函式庫文檔

## All Classes

[BooleanObj](#)  
[CharObj](#)  
[Debug](#)  
[DoubleObj](#)  
[E](#)  
[FloatObj](#)  
[GetFileName](#)  
[ICEBinaryInputStream](#)  
[ICEBinaryOutputStream](#)  
[ICEInputStream](#)  
[ICEOutputStream](#)  
[IntObj](#)  
[L](#)  
[LongObj](#)  
[Ltype](#)  
[SexpException](#)  
[StringObj](#)  
[SunMicroFileFilterExample](#)

[Package](#) [Class](#) [Use](#) [Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV PACKAGE](#) [NEXT PACKAGE](#)

[FRAMES](#) [NO FRAMES](#)

## Package CYICE

### Class Summary

<a href="#">BooleanObj</a>	
<a href="#">CharObj</a>	
<a href="#">Debug</a>	Debug : A class providing two debugging utilities : Trace() and Assert().
<a href="#">DoubleObj</a>	
<a href="#">E</a>	
<a href="#">FloatObj</a>	
<a href="#">GetFileName</a>	
<a href="#">ICEBinaryInputStream</a>	An ICEBinaryInputStream just packages a DataOutputStream.
<a href="#">ICEBinaryOutputStream</a>	An ICEBinaryOutputStream just packages a DataOutputStream.
<a href="#">ICEInputStream</a>	ICEInputStream : A class providing Read functions that return different data types.
<a href="#">ICEOutputStream</a>	An ICEOutputStream just packages a PrintStream.
<a href="#">IntObj</a>	
<a href="#">L</a>	
<a href="#">LongObj</a>	
<a href="#">StringObj</a>	
<a href="#">SunMicroFileFilterExample</a>	A convenience implementation of FileFilter that filters out all files except for those type extensions that it knows about.

### Enum Summary

<a href="#">Ltype</a>	
-----------------------	--