

OOP的基本觀念


CSIE105A

Sheep

TA/SA

SOFTWARE TERMINOLOGY

Then	Now
application	app
program	app
operating system	app
script	app
shell	app
batch file	app
compiler	app
daemon	app
service	app
game	app
patch	app
software	app



目錄

Chap.0	基礎OOP前置複習
Chap.1	繼承與多型
Chap.2	隱私與關係
Chap.3	迭代與遞迴
Chap.4	內部類別
Chap.5	實作多載
Chap.6	抽象類別
Chap.7	其他




sheep's class

Chap.0 基礎OOP前置複習

SOFTWARE TERMINOLOGY

Then	Now
application	app
program	app
operating system	app
script	app
shell	app
batch file	app
compiler	app
daemon	app
service	app
game	app
patch	app
software	app



Java程式介紹

Java是一種物件導向的高階語言

所以在撰寫程式的時候，往往都要宣告類別
而類別內部可以撰寫函式進行想要的工作

類別

函式

變數



sheep's class

Java程式介紹

在Java中，要撰寫主程式需要擁有主要類別與函式

主要類別

```
{  
    主要函式(.....)  
    {  
        .....  
    }  
}
```



Java程式介紹

這個意思表示 公開的主要類別名稱為Main

```
public class Main  
{  
    .....  
}
```



Java程式介紹

Static表示靜態，如果執行其他函式並不會影響主程式執行的順序，會先存在記憶體當中之後繼續執行。

```
public class Main
```

```
{
```

公開 靜態 沒有回傳值的主程式

```
public static void main(String[] args)
```

```
{
```

```
.....
```

```
}
```

```
}
```



sheep's class

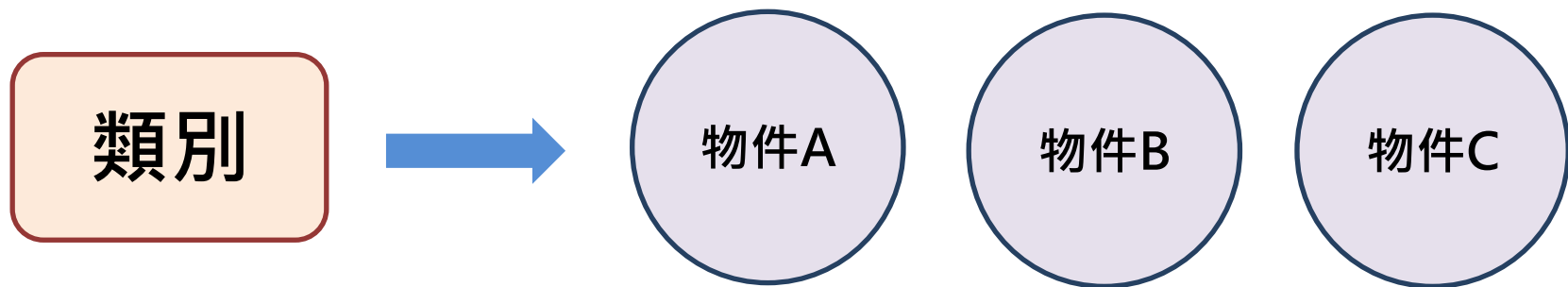
類別與設計

其實類別最核心的就是設計
該如何去設計一個類別才可使程式看起來更完美
更容易讓大家看懂程式到底是在做什麼
尤其是高階語言中的物件導向概念更是重要
在設計類別前，可先好好思考如何設計
設計的好，就是漂亮的程式



類別與設計

○○ P 強調設計類別以產生相似特性的物件



sheep's class

類別與設計

假如你要設計的是糖果，那你應該思考它的特性
他有：顏色、內含物、形狀、價格等等
經過這麼多分析後，你就可以開始設計了！



類別與設計

設計一個屬於自己的糖果

```
class Candy //通常類別第一個字大寫
{
    String color          = "red" ;
    String include        = "chocolate" ;
    String shape          = "ball" ;
    int price = 5;
}
```



sheep's class

類別與設計

設計完後，就可以在主要函數產生糖果了！

```
Candy candy = new Candy();
```

我要知道糖果的顏色，我就：

```
String c = candy.color; // 點，代表的
```

我想知道價格，我就：

```
int p = candy.price;
```

是不是很简单呢？



類別與設計

你也可以修改物件內的變數

```
Candy candy = new Candy();  
candy.color = "blue" ;  
candy.price = 1;
```



sheep's class

類別與設計

你也可以自行生成好多個物件

```
Candy candy1 = new Candy();
```

```
Candy candy2 = new Candy();
```

.....

但是這些生成出來的物件，特性都是一樣的

每產生一次物件就要寫很多行來修改

是不是很麻煩？



sheep's class

建構值的設計

於是建構函數就這麼出現了！

Constructor （建構值 / 建構元 / 建構函數）

只要每產生出物件，建構值就會馬上被呼叫

它主要在程式中，是擔任初始化的角色

經過建構值的初始化，物件才會正式被產生



sheep's class

建構值的設計

事實上，沒有寫建構值的類別
程式會自動補上空的可建構值

```
class Candy
{
    String color           = "red" ;
    String include         = "chocolate" ;
    String shape           = "ball" ;
    int price = 5;
    public Candy() { }
}
```



sheep's class

建構值的設計

撰寫建構值的時候，需思考哪些變數需要被初始化

例如：設計的糖果只有顏色和形狀會不相同

那你的建構值就只需要放入這 2 個進行初始化

```
public Candy(String c,String s)
```

```
{  
    color = c;  
    shape = s;  
}
```



sheep's class

建構值的設計

最適當的寫法如下：

```
public Candy(String color,String shape)
{
    this.color  = color;
    this.shape = shape;
}
```

this表示類別裏面的變數



建構值的設計

當你撰寫完建構值時，空的建構值就會馬上消失
如果你希望無參數的建構值也要用到的話
可以呼叫已經寫好的建構值，是不是更方便呢？：

```
public Candy()  
{  
    this( "black" ," square" );  
}  
public Candy(String color,String shape)  
{  
    this.color  = color;  
    this.shape = shape;  
}
```



建構值的設計

於是糖果的程式碼幾乎是完成了

```
class Candy
{
    String color          = "red" ;
    String include        = "chocolate" ;
    String shape          = "ball" ;
    int price             = 5;
    public Candy()
    {
        this( "black" ," square" );
    }
    public Candy(String color,String shape)
    {
        this.color  = color;
        this.shape  = shape;
    }
}
```



sheep's class

建構值的設計

主程式可依照自己的需求生產出物件

若不知道要給什麼條件，無須提供參數給建構值：

```
Candy candy1 = new Candy();
```

若已經知道要給予條件，就直接打入建構值內：

```
Candy candy2 = new Candy( "red" , " ball" );
```



函數的設計

上面已經介紹完建構函數了
其實你也可以在類別中撰寫其他函數
例如：從1加到N的函數.....



函數的設計

設計→從1加到N的函數

```
public static int sumToN(int N)
{
    int sum = 0;
    for(int i=1;i<=N;i++)
        sum+=i;
    return sum;
}
```

在主要類別內才需要加static



sheep's class

函數的設計

設計→從1加到N的函數

參數

```
public static int sumToN(int N)
```

```
{  
    回傳值資料型態
```

```
    int sum = 0;
```

```
    for(int i=1;i<=N;i++)
```

```
        sum+=i;
```

```
    return sum;
```

```
}  
    回傳的變數
```



sheep's class

函數的設計

我們可以為Candy再設計函數

```
void addPrice(int n)
```

```
{  
    price += n;
```

```
}  
void addPrice()
```

```
{  
    price += 10;  
}
```



sheep's class

函數的設計

```
class Candy                //類別更具完整性了
{
    String color            = "red" ;
    String include          = "chocolate" ;
    String shape            = "ball" ;
    int price                = 5;
    public Candy()
    {
        this( "black" ," square" );
    }
    public Candy(String color,String shape)
    {
        this.color = color;
        this.shape = shape;
    }
    void addPrice(int n)
    {
        price+=n;
    }
    void addPrice()
    {
        price+=10;
    }
}
```



sheep's class

多載

你一定會問，為什麼可以新增 2 個相同名稱的函數
因為程式提供多載 (Overloading) 的功能
同樣的函數提供多種的參數形式
可以根據自己的需求，來呼叫要用的形式

```
void addPrice(int n){...}  
void addPrice(){...}
```



多載

舉例來說：

1 >

```
int n = sc.nextInt(); //輸入十進位的整數
```

2 >

```
int r = 8;                //輸入基底
```

```
int n = sc.nextInt(r); //將r進位轉成十進位後放入
```



sheep's class

多載

在oracle官方提供的API可查詢類別中多載的函數
以剛剛的例子來說，可以查到：

回傳值

int

int

函數

nextInt()

Scans the next token of the input as an int.

nextInt(int radix)

Scans the next token of the input as an int.



sheep's class

物件的使用

請注意！物件與基本資料型態的使用方式並非相同
舉例：

1 > 基本資料型態

```
Int a = 10;
```

```
Int b = 12;
```

```
a = b;
```

```
b = 50;
```

此時，a是12，b是50



物件的使用

2 > 物件 (Reference)

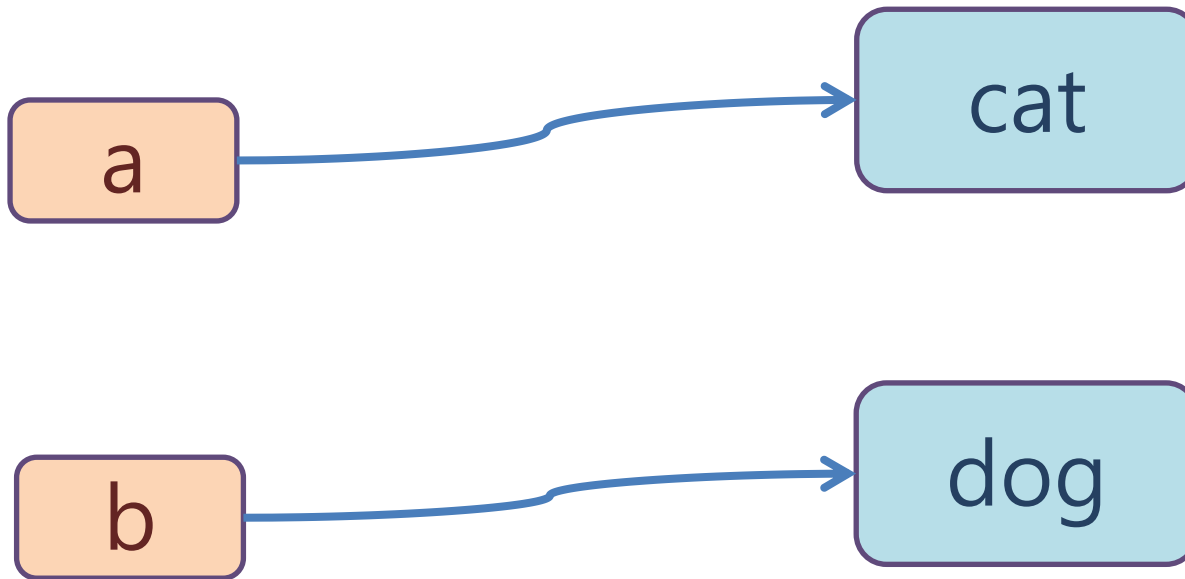
```
String a = "cat" ;  
String b = "dog" ;  
a = b;  
b = "mouse" ;
```

此時，不可將物件的值作為一般的解讀
須根據參考位置來做判斷
(之後會提到字串並非基本資料型態)



References

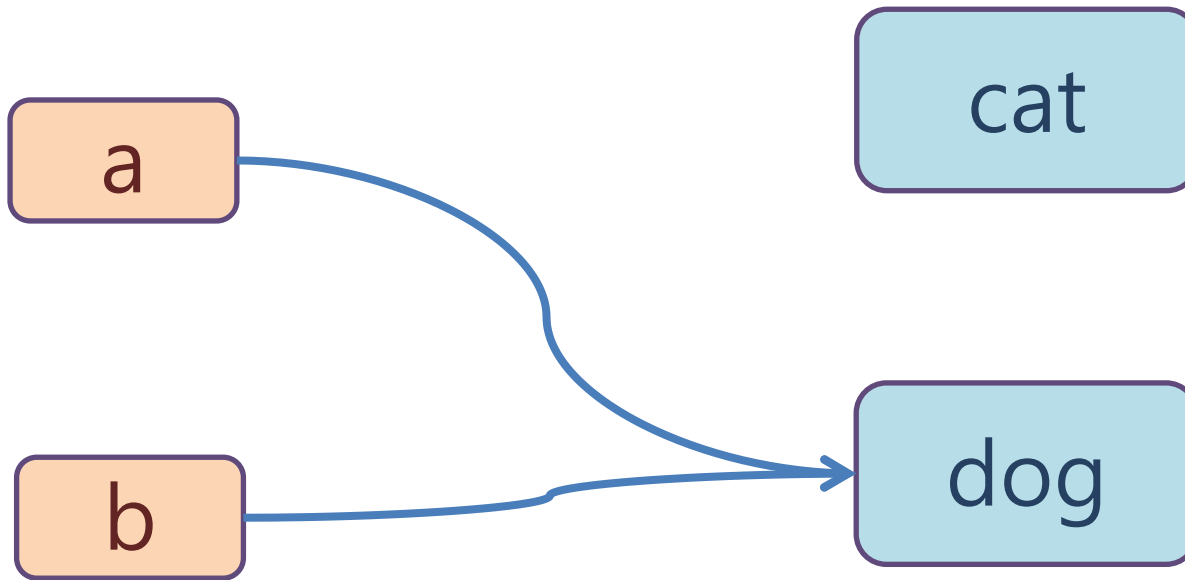
第一步



sheep's class

References

第二步 b將b指向的位址給a
(即References參考位置)



References

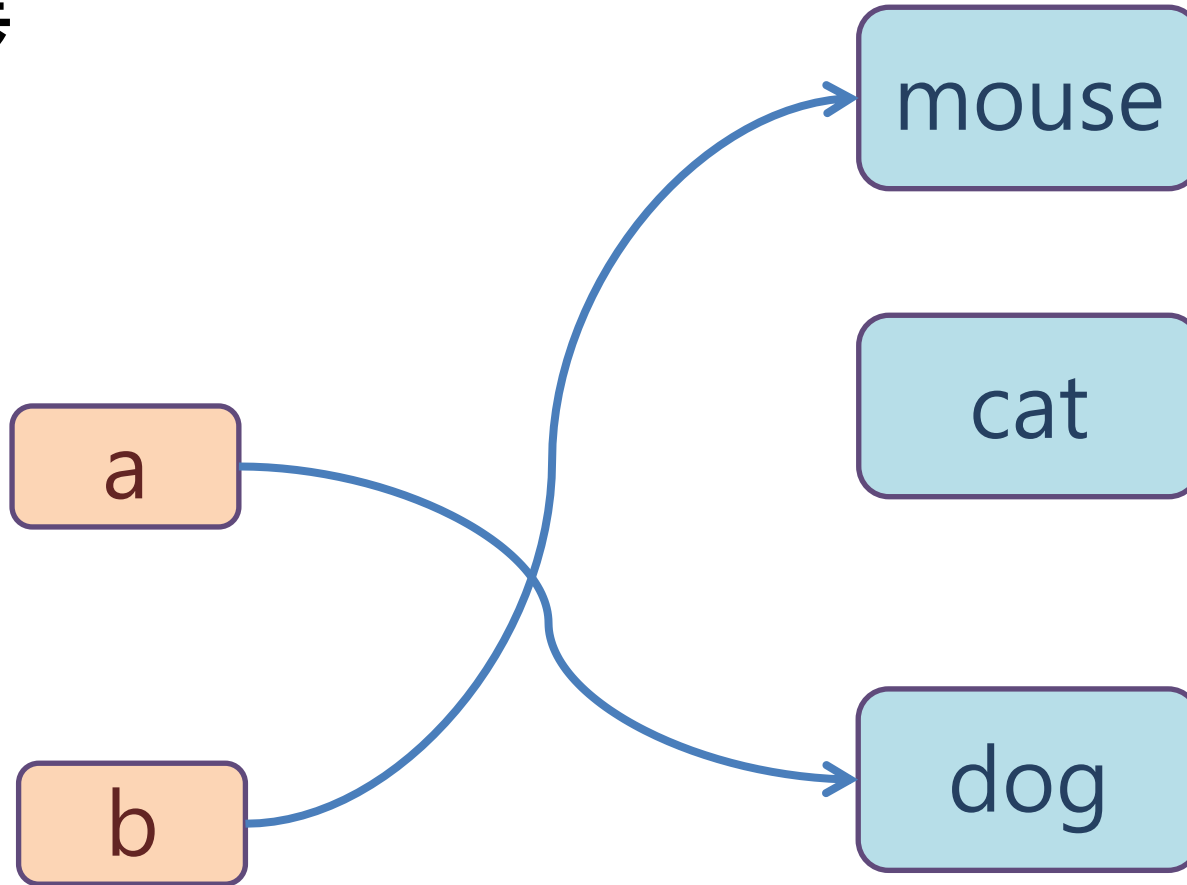
第三步 `b = "mouse"` 不會是將dog的值改掉
而是程式new一個String的物件叫mouse
然後才會將mouse的References傳給b，指向它



sheep's class

References

第三步

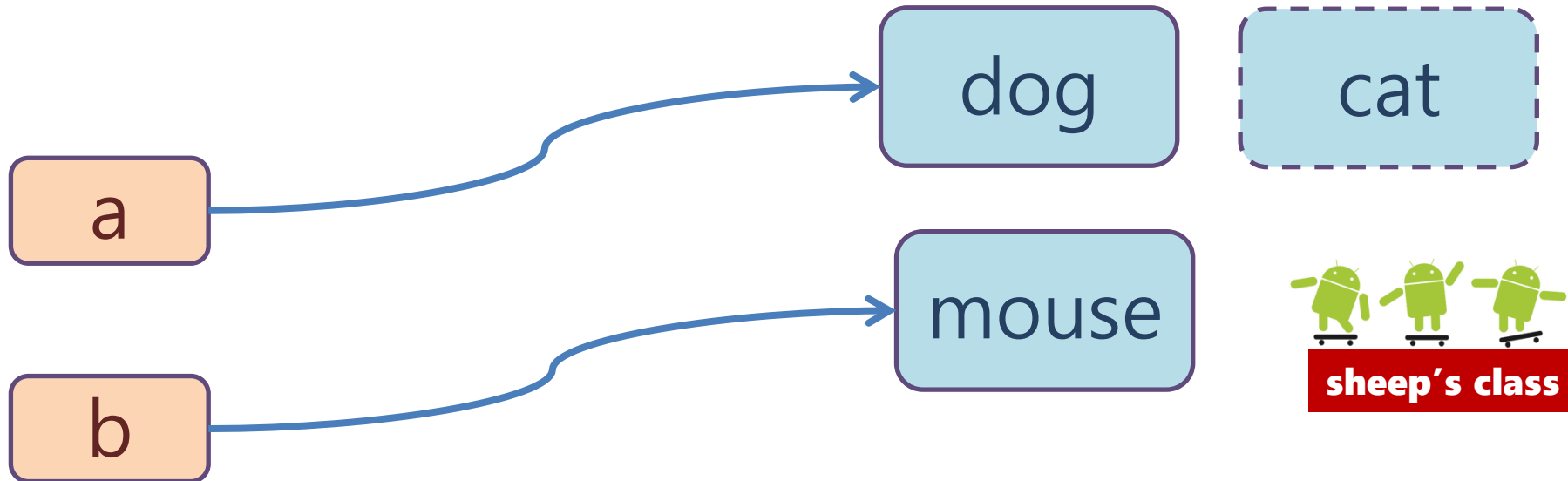


sheep's class

References

第四步 而cat將會是被懸浮的物件 之後會被Java垃圾車自動載走 (Garbage collection)

Note : 並非所有程式語言都擁有這樣的機制
像C語言就要自己手動分配，否則空間會爆炸



References

不懂程式語言的特性可能會覺得上面的事情很神奇
但是只要知道Java具有References的特性
以後發生類似的狀況就能很快的解讀出狀況
接下來就來介紹程式在呼叫函數時的其他特性



字串

字串可能在你學習變數宣告的時候就學到了
但是你一定會有一些疑惑：

- 1 > 為什麼字串的S要大寫？
- 2 > 為什麼不能當一般變數使用？
- 3 > 為什麼宣告字串會用到new？

.....

接下來將會從頭開始介紹字串



變數差異

基本資料型態 (primitive data type)

例如：int、double、char.....

物件資料型態 (References)

例如：String、Integer、Double

沒錯，String不是基本資料型態



字串宣告

字串的標準宣告方式：

```
String s = new String( "" );
```

看到new，就是根據建構值的建構後，產生物件



字串宣告

用物件的時候，盡量避免只用運算子 **=** 來初始化

```
String s = "" ;
```

注意對物件來說

等號相當於改變參考位址

所以指向的物件會改變，而非改變原本的值



sheep's class

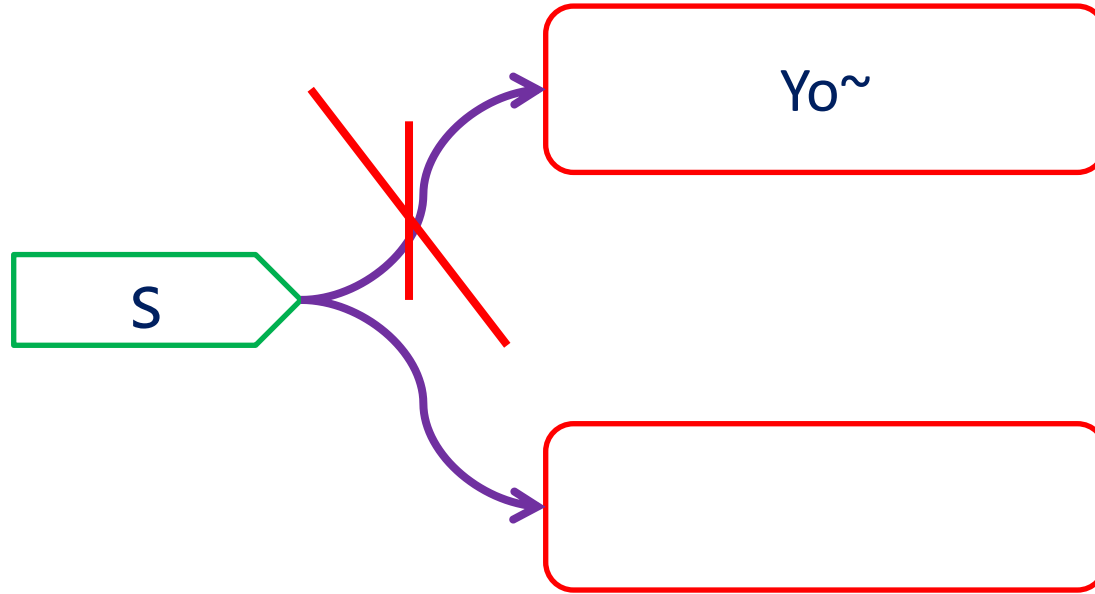
字串的資料

如果你想更改字串內的資料
你只能另外產生一個物件，並將參考位址指向它

```
String s = new String( "Yo~" ); //已經產生  
s = new String( "" ); //改指標同於改資料
```



字串的資料



懸浮的物件會被Java垃圾車載走

字串的資料

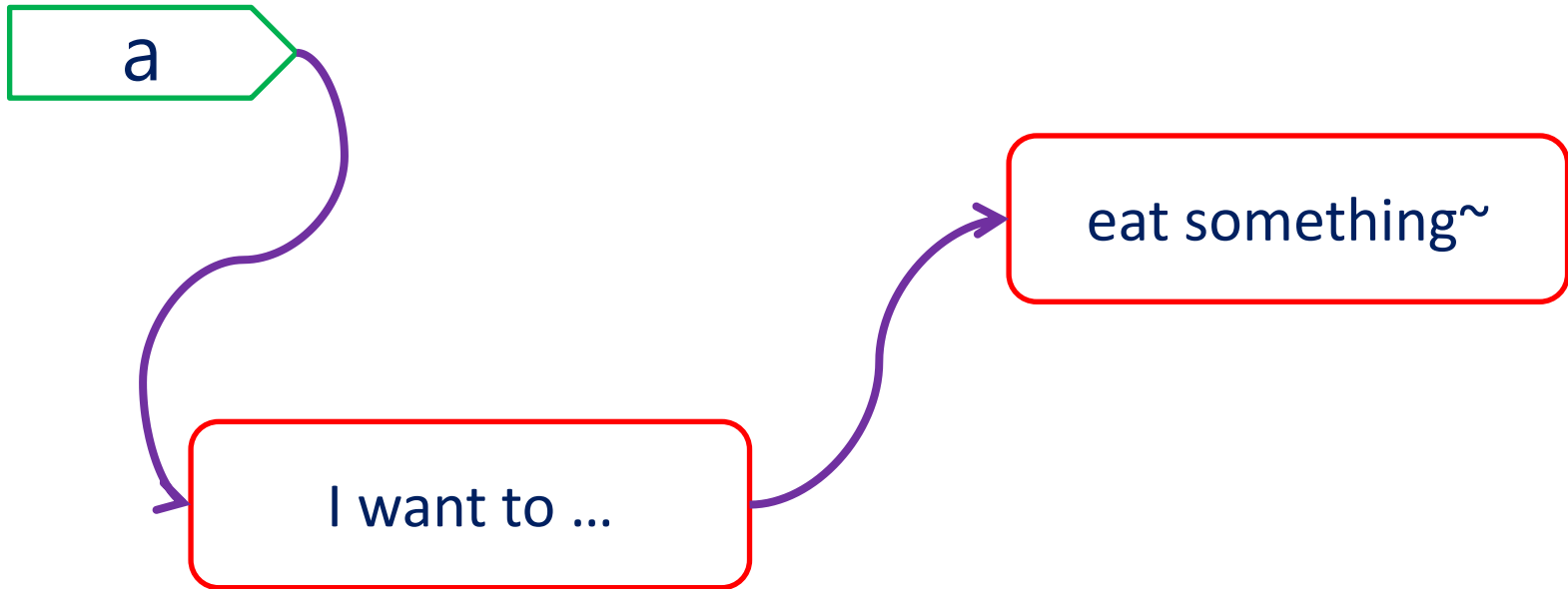
對字串來說，用運算子 **+** 就相當於是鏈結在一起
例如：

```
String a = new String( "I want to ... " );  
a = a + " sleep" ;
```

這樣a會變成：I want to ... sleep
事實上是鏈結，也並非改內部的值.....



字串的資料



字串的資料

對字串來說，用運算子 **+** 就相當於是鏈結在一起
例如：

```
String a = new String( "I want to " );
```

```
a = a + " ... " ;
```

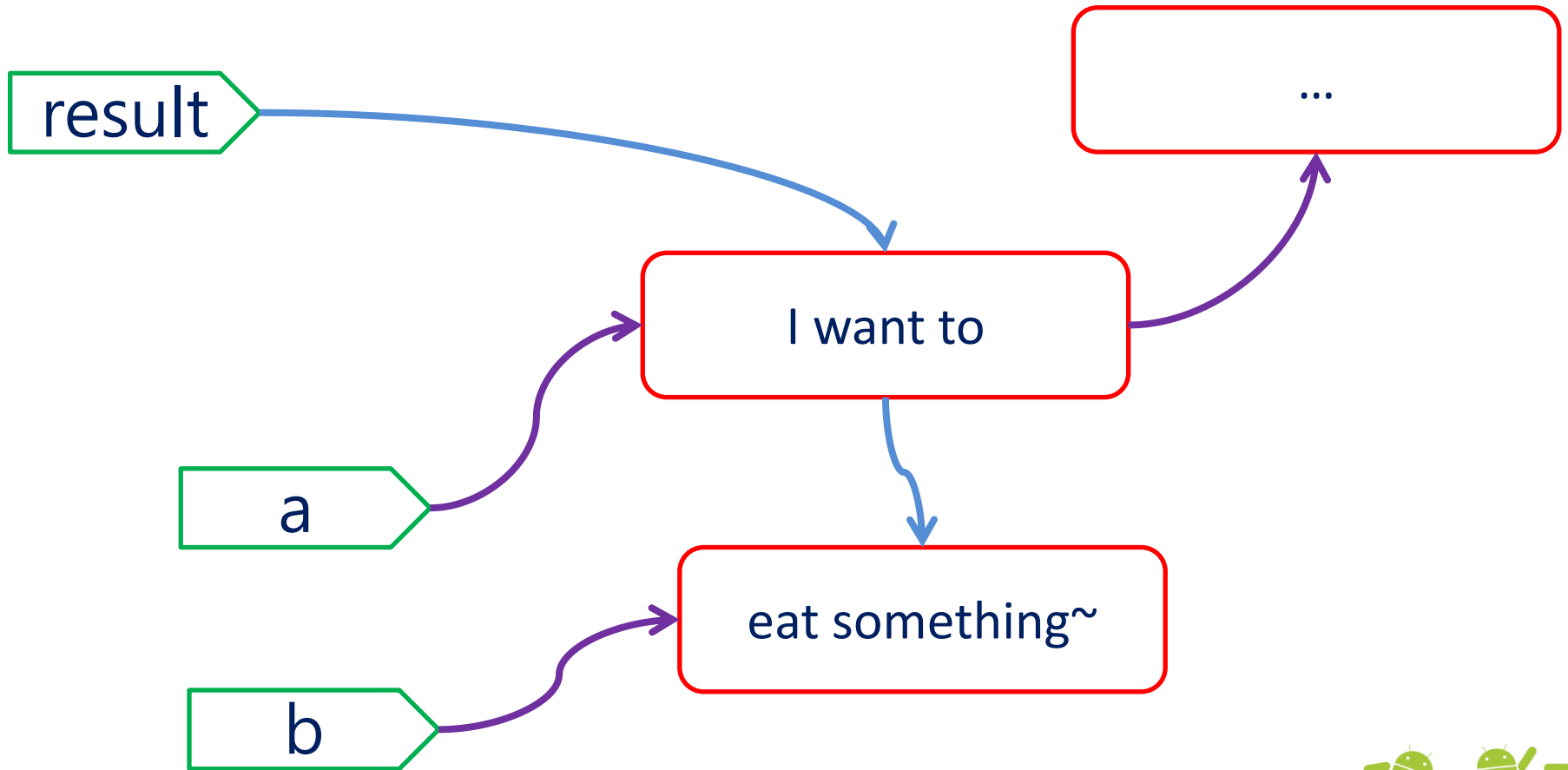
```
String b = new String( "eat something~" );
```

```
String result = a+b;
```

這樣a和b就合在一起是一句了



字串的資料



sheep's class

因為都是獨立的，所以互相不會影響到彼此的值

字串的比較

你知道嗎？要如何判斷 2 個字串是否相同？

舉例：

```
String s1 = "123" ;
```

```
String s2 = "123" ;
```

謎：我知道我知道～

```
    if(s1==s2)
```

```
        System.out.println( "相等" );
```

```
    else
```

```
        System.out.println( "不相等" );
```



字串的比較

```
String s1 = "123" ;  
String s2 = "123" ;  
if(s1==s2)  
    System.out.println( "相等" );  
else  
    System.out.println( "不相等" );
```

上面會輸出什麼呢？是相等嗎？

答案是：不相等 (什麼！？)



字串的比較

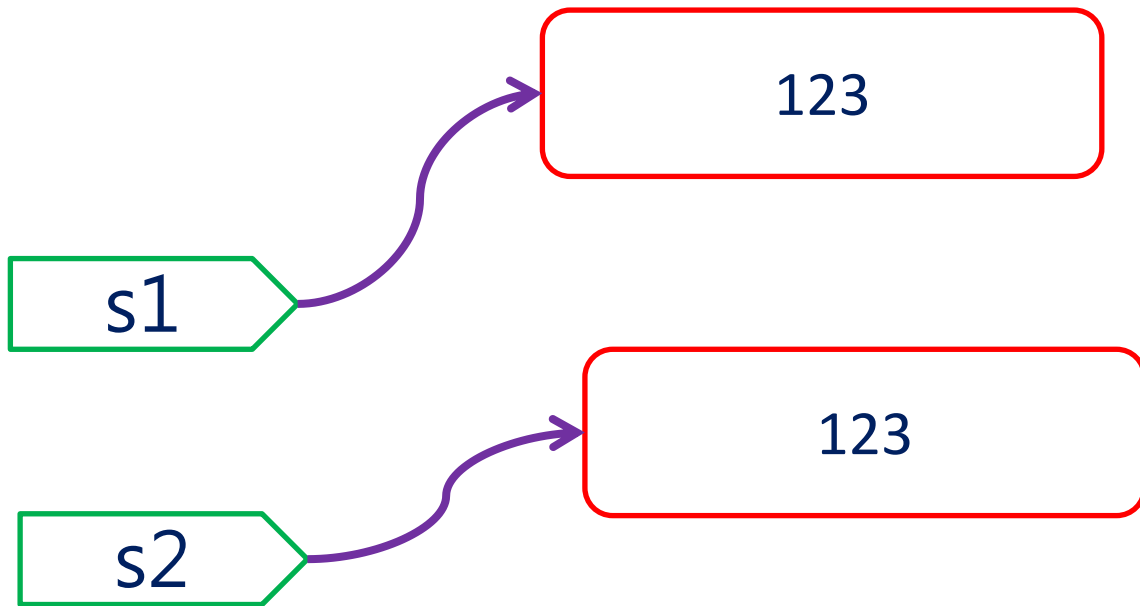
原因很簡單，如果你已經有上面Reference的一些背景知識熟悉的話，大概知道為什麼對物件來說，運算子的運算都是跟Reference相關所以你比較的是什麼？知道了嗎？

答案在下一頁喔，再想一下～



字串的比較

因為運算子等號他比較的是物件的參考位址
也就是說，這 2 個物件是不是指向同一個參考位址
但是很明顯的，這 2 個物件都是額外new出來的
所以是 2 個不相關的物件～



字串的比較

那 . . . 要怎麼比較 2 個字串是否相等呢？

你可以：

- 1 > 跑迴圈取出字元慢慢比較
- 2 > 用API的method `equals()`



sheep's class

字串的比較

1 > 跑迴圈取出字元慢慢比較

```
//確保不會拋出IndexOutOfBoundsException異常
int size = Math.min(s1.length(),s2.length());
boolean isEqual = true;
for(int i=0;i<size;i++){
    if(s1.charAt(i)!=s2.charAt(i)){
        isEqual = false;
        break;
    }
}
if(isEqual)
    System.out.println( "相等" );
else
    System.out.println( "不相等" );
```



字串的比較

2 > 用API的method equals()

```
if(s1.equals(s2))  
    System.out.println( "相等" );  
else  
    System.out.println( "不相等" );
```

是不是很快呢？



字串的method

Java API提供字串很多不錯的method可以用
比較重要一定要知道的如下：

回傳型態	方法名稱與描述
char	charAt(int index)：取出字串中的第i個字元
boolean	contains(CharSequence s)：該字串中是否包含字串s
boolean	equals(Object o)：判斷兩個字串是否相等
int	length()：回傳字串的長度
String	substring(int begin,int end)：回傳子字串
int	indexOf(String s)：尋找找到的字串，回傳索引值 但是只會回傳先找到的字串，找不到就回傳-1



字串的疑問

你一定會問，可是我如果**必須**要更改字串內的資料該怎麼做呢？

接下來，讓我來跟你鄭重的介紹

StringBuffer與StringBuilder



字串家族

名稱		推出時間	執行速度
String	字串	Java SE2	慢
StringBuffer	字串緩衝	Java SE2	中
StringBuilder	字串建構者	Java SE5	快

(單執行緒時)

StringBuffer與StringBuilder的優點在
他不僅可以增加與修改字串
而且速度也比String快很多



字串建構者

StringBuilder使用方式與字串差不多
只是他的型態不是String，而是StringBuilder
所以你不能：

```
StringBuilder sb = "123" ;  
StringBuilder sb = new String( "123" );
```

你要這樣：

```
StringBuilder sb = new StringBuilder( "123" );
```



字串建構者

要加字串、整數、字元等變數 需使用method

例如：

```
StringBuilder sb = new StringBuilder( "" );  
sb.append(123);  
sb.append( "yo~" );
```



字串建構者

如果要刪除第?個字元，可以用：

```
StringBuilder sb = new StringBuilder( "" );  
sb.append(123456);  
sb.deleteCharAt(0);  
sb.deleteCharAt(2);
```

裡面放索引值



字串建構者

如果你希望讓某一個字串反轉過來
其實你可以借助StringBuilder的method輕鬆辦到

```
String s = "12345";  
StringBuilder sb = new StringBuilder(s);  
sb.reverse();  
s = sb.toString();    // StringBuilder轉成String
```



字串建構者的method

StringBuilder在字串修改上，扮演很重要的角色
以下是較重要的method：

回傳型態	方法名稱與描述
StringBuilder	append(?)：加入資料於後面
StringBuilder	insert(int offset, ?)：從哪裡插入資料
StringBuilder	delete(int start,int end)：刪除一個片段
StringBuilder	deleteCharAt(int index)：刪除第幾個字元
StringBuilder	reverse()：反轉整個StringBuilder
String	substring(int begin,int end)：回傳子字串
String	toString()：轉成字串



字串建構者的method

StringBuilder也有和String相同的method可以用


回傳型態	方法名稱與描述
char	charAt(int index)：取出字串中的第i個字元
boolean	contains(CharSequence s)：該字串中是否包含字串s
boolean	equals(Object o)：判斷兩個字串是否相等
int	length()：回傳字串的長度
String	substring(int begin,int end)：回傳子字串
int	indexOf(String s)：尋找找到的字串，回傳索引值 但是只會回傳先找到的字串，找不到就回傳-1



Chap.1 繼承與多型

SOFTWARE TERMINOLOGY

Then	Now
application	app
program	app
operating system	app
script	app
shell	app
batch file	app
compiler	app
daemon	app
service	app
game	app
patch	app
software	app



繼承與多型

前面提到之前學的物件導向基本觀念
也許你還是會覺得頭昏腦脹，到底該如何學下去
其實不急，程式接著學往往可以跟著複習之前的
當你走到想回頭的時候，千萬別輕易往回看
你會被後面跟著你的蛇女石化的（X



繼承與多型

OOP最重要的三元素由高到低分別是

- 1> 封裝
- 2> 繼承
- 3> 多型



sheep's class

繼承與多型

程二中學到了基本的物件封裝的概念
現在將會告訴大家
物件之間是可以有繼承關係的
物件也可以有多種型態呈現



sheep's class

繼承

當 A 類別宣告繼承了 B 類別時
A 類別就會成為 B 類別的子類別
B 類別則成為了 A 類別的父類別



sheep's class

繼承

當 A 類別宣告繼承了 B 類別時
A 類別就會成為 B 類別的子類別
B 類別則成為了 A 類別的父類別

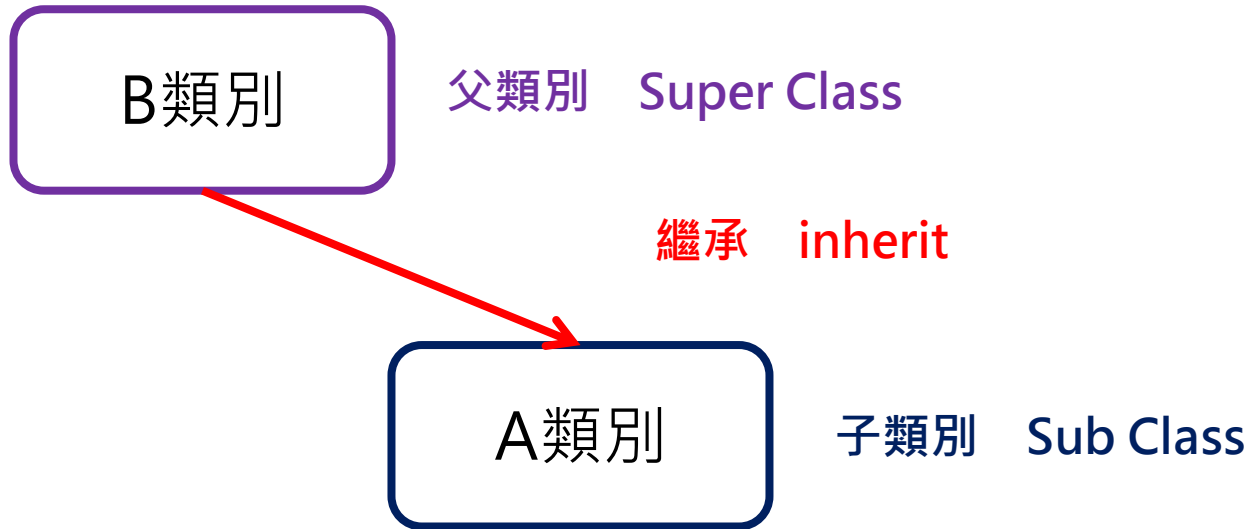


sheep's class

繼承

程式可以表示成：

```
class A extends B{.....}  
class B{.....}
```



sheep's class

繼承

什麼時候會用到繼承呢？

當你要用這個類別去產生相似特性的類別時

例如：水可以做出咖啡、果汁

那咖啡和果汁可以**繼承了水的特性**

```
class 水{.....}
```

```
class 咖啡 extends 水{.....}
```

```
class 果汁 extends 水{.....}
```



sheep's class

繼承

繼承的類別，內涵的變數、函數.....都會繼承下來
當沒有宣告它的隱私型態時，就會直接可以被使用
但宣告成private時，繼承下來的東西是不能用的

下一章將會介紹隱私型態



多型

現在要介紹的是多型的概念

多型的意思，就是多種型態的意思

例如：水果可以是香蕉、蘋果、橘子.....

所以那些就是水果的多型

形狀可以是圓形、長方形、三角形.....

所以那些就是形狀的多型



sheep's class

多型

水果的多型：

```
class Fruit{.....}
```

```
class Banana extends Fruit{.....}
```

```
class Apple extends Fruit{.....}
```

```
class Orange extends Fruit{.....}
```

當從父類別繼承下來的成員

可以依照自己的需要選擇要不要覆寫



sheep's class

多型

典型的題目，形狀的多型：

```
class Shape{.....}
```

```
class Rectangle extends Shape{.....}
```

```
class Triangle extends Shape{.....}
```

```
class Circle extends Shape{.....}
```



sheep's class

this 與 super

當類別要呼叫自己的變數時，可以使用This

```
class Paper
{
    String text;
    public Paper ()
    {
        this.text = "Hi~";
    }
}
```



sheep's class

this 與 super

當類別要呼叫父類別的變數時，可以使用super

```
class Card extends Paper
```

```
{
```

```
    public Card()
```

```
    {
```

```
        super.text = "Happy Birthday! ";
```

```
    }
```

```
}
```



sheep's class

Chap.2 隱私與關係

SOFTWARE TERMINOLOGY

Then	Now
application	app
program	app
operating system	app
script	app
shell	app
batch file	app
compiler	app
daemon	app
service	app
game	app
patch	app
software	app



隱私型態

類別有這三種隱私型態，當然也可以用在變數上

Private	私有的
Public	公開的
Protected	被保護的

e.g. :

```
public class Main{.....}  
private int number = 100;
```



隱私型態

Private

只有我自己可以用

Public

大家都可以用

Protected

只有跟我有血緣關係的人才可以用



OOP的關係

is-a關係

全名叫is a kind of

當 2 個類別彼此有關係的話（類別cake和bread）
我們可以唸做 **Cake is a kind of Bread**
一旦有這個關係，就表示有繼承的涵義
唸作**蛋糕也是麵包的一種**



OOP的關係

is-a關係 全名叫is a kind of

class Cake{.....}

class Cake extends Bread{.....}

可記成 subclass is-a superclass



OOP的關係

has-a關係 全名叫A has a B

組合的關係存在 (whole/parts)
簡單來說就是A是由B所組成的
當兩個類別分別是GreenTea和Water

```
class Water{.....}  
class GreenTea{  
    public Water water; //代表綠茶中包含水  
}
```




sheep's class

Chap.3 迭代與遞迴

SOFTWARE TERMINOLOGY

Then	Now
application	app
program	app
operating system	app
script	app
shell	app
batch file	app
compiler	app
daemon	app
service	app
game	app
patch	app
software	app



迭代與遞迴

這裡要介紹的是各位常用到的迭代法和遞迴法

迭代（iterative）：呼叫方式為Button-Up
通常是大家寫程式常用的方法

遞迴（recursive）：呼叫方式為Top-Down
通常是為了縮短代碼或方便來寫的



迭代法

一般用迭代法都是直接使用迴圈
以 $1+...+N$ 為例：

```
int sum = 0;  
for(int i=1;i<=N;i++)  
    sum+=i;
```

上面就是典型迭代演算法的呈現



遞迴法

遞迴需要寫成function來方便呼叫自己
以 $1+...+N$ 為例：

```
static int f(int n)
{
    if(n==1)
        return 1;
    else
        return n+f(n-1);
}
```



遞迴法

看不懂嗎？沒關係，讓我們來慢慢拆開

假設我主程式呼叫 $f(5)$

則算式（比對上個程式看）：

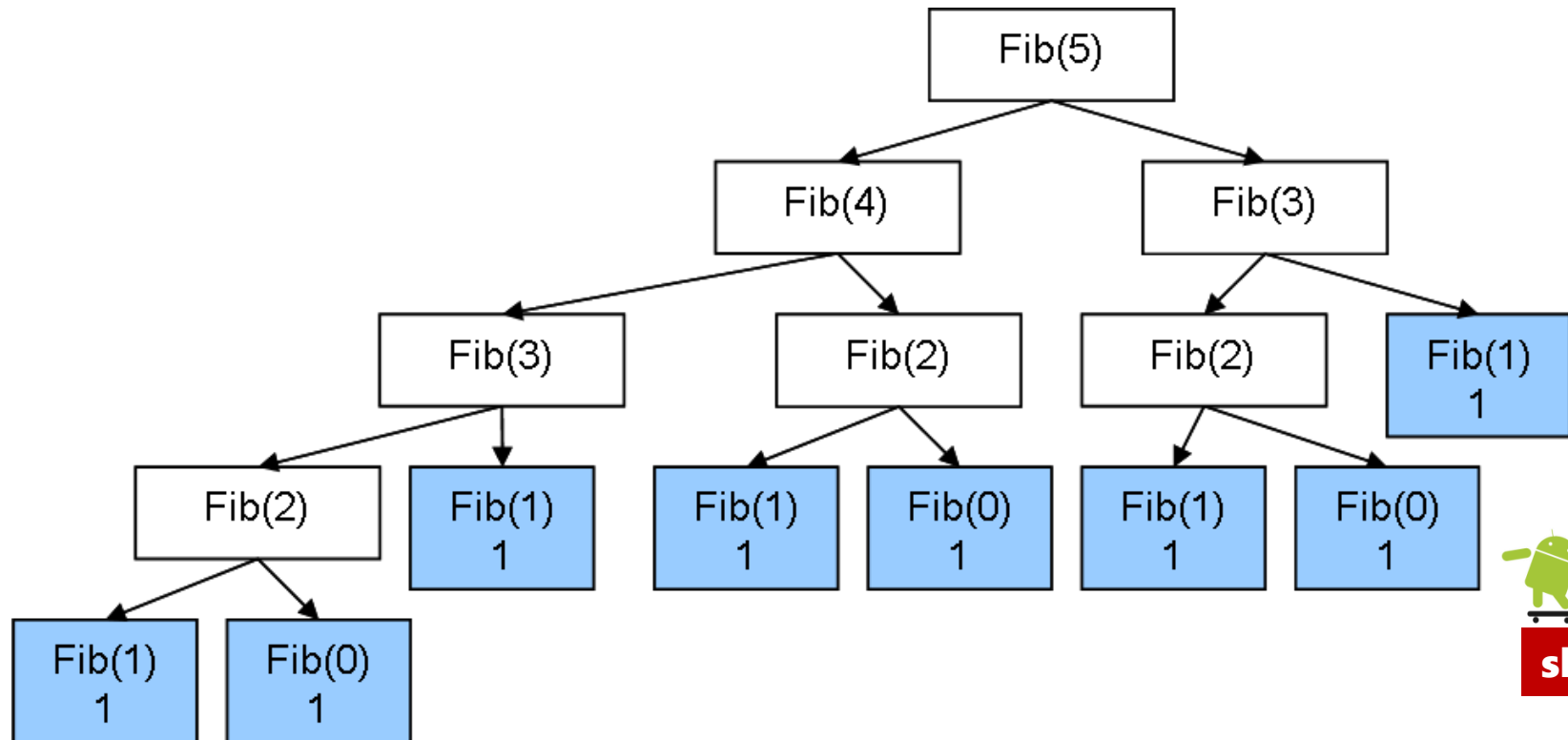
$f(5)$
 $= 5 + f(5-1)$ //因為 $5! = 1$
 $= 5 + (4 + f(3))$ //同理
 $= 5 + (4 + (3 + f(2)))$
 $= 5 + (4 + (3 + (2 + f(1))))$
 $= 5 + (4 + (3 + (2 + (1))))$ // $n=1$ 遞迴中止



sheep's class

遞迴法

遞迴法會由根開始Top-Down產生資料最後回傳
以費式數列來說：



sheep's class

遞迴法

補充：

遞迴需要由上往下再回傳值

所以效率會不太理想

由上頁的圖你會發現fib(1)重複呼叫多次

所以程式的效率更加的差

如果搭配動態規劃法優化的話，可以使效率更好


各位有興趣可以試試看



Chap.4 內部類別

SOFTWARE TERMINOLOGY

Then	Now
application	app
program	app
operating system	app
script	app
shell	app
batch file	app
compiler	app
daemon	app
service	app
game	app
patch	app
software	app



內部類別

可以稱呼

Inner class 內部類別

Nested class 巢狀類別

好處是可以存取外部類別的成員（ e.g. : private ）



sheep's class

內部類別

```
class Point2D
{
    private int x,y;
    .....
    private class Inner
    {
        private int a = x;
        private int b = y;
    }
}
```



sheep's class

Chap.5 實作多載

SOFTWARE TERMINOLOGY

Then	Now
application	app
program	app
operating system	app
script	app
shell	app
batch file	app
compiler	app
daemon	app
service	app
game	app
patch	app
software	app



實作多載

在程式設計二的內容中，有介紹到多載
尤其是Java API中，如何查看有多載的method
在這裡就要學會如何實作多載的method



多載的使用時機 (override)

- 1 > 擴充function
- 2 > 修改function



多載的使用時機 (override)

1 >

在同類別中，需要擁有多種的不同參數表示方式
則需要用到多載

例如：2 個變數相加
可能會遇到很多狀況

正數 + 整數

浮點數 + 整數

字串 + 整數

.....



sheep's class

多載的使用時機 (override)

但是如果只有一個int+int的函數不可能都辦到的
例如：

```
int add(int a,int b) {return a+b;}
```

所以需要寫其他函數來幫忙達成

```
double add(double a,int b) {return a+b;}
```

```
String add(String a,int b) {return a+b;}
```

.....



sheep's class

多載的使用時機 (override)

最方便的地方看出來了嗎？

你不需要取不同的名稱，用同個名稱即可！

這就是多載方便的地方，方便大家開發程式

```
int add(int a,int b) {return a+b;}
```

```
double add(double a,int b) {return a+b;}
```

```
String add(String a,int b) {return a+b;}
```



sheep's class

多載的使用時機 (override)

2 >

當你跟父類別繼承下來的function
不想用他的方法但是又要取同樣名稱時
你可以經由多載的方式改寫父類別繼承下來的內容



多載的使用時機 (override)

```
abstract class Shape           //抽象的父類別
{
    abstract public int Area(); //抽象function
}

class Square extends Shape     //繼承了Shape
{
    int side = 0;

    public Square(int n)
    {
        side = n;
    }

    public int Area()           //多載Area method
    {
        return side * side;
    }
}
```



sheep's class

Chap.6 抽象類別

SOFTWARE TERMINOLOGY

Then	Now
application	app
program	app
operating system	app
script	app
shell	app
batch file	app
compiler	app
daemon	app
service	app
game	app
patch	app
software	app



抽象類別

抽象類別提供一個類別中的成員宣告成抽象的形式
例如：

```
abstract class Shape {  
    abstract public int Area();  
}
```

內部的成員皆可宣告成**abstract**的抽象型態



sheep's class

抽象類別


你一定會問說，到底有什麼用呢？
當你的中心是在設計子類別，而非父類別時
父類別這個時候，只需設計出一個雛形來
以方便子類別的設計，那麼這時候就會用到它了
通常用在方便快捷設計時，會用上



Chap.6 其他

SOFTWARE TERMINOLOGY

Then	Now
application	app
program	app
operating system	app
script	app
shell	app
batch file	app
compiler	app
daemon	app
service	app
game	app
patch	app
software	app



toString的實作

當你要檢視物件中的資料時
toString會是很方便的東西
例如有一個類別是存放一個點座標的：

```
class Point2D{  
    int x;  
    int y;  
    Point2D(int x,int y){  
        this.x=x;  
        this.y=y;  
    }  
}
```



sheep's class

toString的實作

但是你主程式直接把產生的物件print出來時
會發現他print出來的是一組hashcode，而非座標

```
Point2D a = new Point2D(1,3);
```

```
System.out.println(a);
```

輸出：

```
Point2D@232204a1
```

所以你需要實作toString來顯示座標的值



toString的實作

在Java中產生的任何類別，都會自動繼承Object
所以事實上，**class Point2D extends Object{...}**
Object中，包含了toString()這個函數
所以我們要Override這個toString()函數



toString的實作

可以在toString()中，改成自己想呈現的字串

```
public String toString()  
{  
    return "("+x+", "+y+")";  
}
```

注意：回傳型態和隱私型態必須相同才可override



sheep's class

toString的實作

最後會發現輸出變成自己想要的值：
(1,3)

