|  |
| --- |
| 資訊工業策進會 數位教育研究所 |
| 初級行動裝置程式設計師能力鑑定 考前輔導研習營 |
| **行動裝置程式開發-Android程式設計 程式設計概論** |
|  |
| 林俊昌 |
|  |

目 錄

[1 程式語言基礎 5](#_Toc497772073)

[1.1 Java資料型別(Data type) 5](#_Toc497772074)

[1.1.1 基本資料型別(Primitive data type) 5](#_Toc497772075)

[1.1.1.1 隨堂測驗 5](#_Toc497772076)

[1.1.2 參考資料型別(Reference data type) /類別(Class) 5](#_Toc497772077)

[1.2 運算子(Operator) 5](#_Toc497772078)

[1.2.1 常用運算子 5](#_Toc497772079)

[1.2.2 運算子優先順序 6](#_Toc497772080)

[1.2.3 隨堂測驗 6](#_Toc497772081)

[1.3 控制結構(Control structure) 9](#_Toc497772082)

[1.3.1 重複結構(Repetition structure) 9](#_Toc497772083)

[1.3.1.1 重點摘要 9](#_Toc497772084)

[1.3.1.2 隨堂測驗 10](#_Toc497772085)

[1.3.2 選擇結構(Selection structure) 13](#_Toc497772086)

[1.3.2.1 重點摘要 13](#_Toc497772087)

[1.3.2.2 隨堂測驗 14](#_Toc497772088)

[1.4 變數有效範圍(Scope) 17](#_Toc497772089)

[1.4.1 主要分類 17](#_Toc497772090)

[1.4.1.1 實體變數(Instance variable) 17](#_Toc497772091)

[1.4.1.2 類別變數(Class variable) 17](#_Toc497772092)

[1.4.1.3 區域變數(Local variable) 17](#_Toc497772093)

[1.4.1.4 區塊變數(Block variable) 17](#_Toc497772094)

[1.4.2 隨堂測驗 18](#_Toc497772095)

[1.5 識別名稱(Identifier) 19](#_Toc497772096)

[1.5.1 Java識別名稱的基本規則 19](#_Toc497772097)

[1.5.2 隨堂測驗 19](#_Toc497772098)

[1.6 函式/方法(Function/Method) 20](#_Toc497772099)

[1.6.1 重點摘要 20](#_Toc497772100)

[1.6.2 隨堂測驗 21](#_Toc497772101)

[1.7 例外處理(Exception handling) 21](#_Toc497772102)

[1.7.1 重點摘要 21](#_Toc497772103)

[1.7.2 隨堂測驗 22](#_Toc497772104)

[1.8 逃逸序列(Escape sequence) 23](#_Toc497772105)

[1.8.1 常用逃逸序列 23](#_Toc497772106)

[1.8.2 隨堂測驗 23](#_Toc497772107)

[1.9 註解(Comment) 24](#_Toc497772108)

[1.9.1 Java提供之註解 24](#_Toc497772109)

[1.9.2 隨堂測驗 24](#_Toc497772110)

[2 物件導向程式語言 24](#_Toc497772111)

[2.1 物件(Object)導論 25](#_Toc497772112)

[2.1.1 重點摘要 25](#_Toc497772113)

[2.1.2 隨堂測驗 25](#_Toc497772114)

[2.2 繼承(Inheritance) 26](#_Toc497772115)

[2.2.1 重點摘要 26](#_Toc497772116)

[2.2.2 隨堂測驗 27](#_Toc497772117)

[2.3 封裝(Encapsulation) 29](#_Toc497772118)

[2.3.1 重點摘要 30](#_Toc497772119)

[2.3.2 隨堂測驗 30](#_Toc497772120)

[2.4 建構子(Constructor) 31](#_Toc497772121)

[2.4.1 重點摘要 31](#_Toc497772122)

[2.4.2 隨堂測驗 32](#_Toc497772123)

[2.5 覆寫(Override) 33](#_Toc497772124)

[2.5.1 重點摘要 33](#_Toc497772125)

[2.5.2 隨堂測驗 34](#_Toc497772126)

[2.6 方法多載(Method overloading) 35](#_Toc497772127)

[2.6.1 重點摘要 35](#_Toc497772128)

[2.6.2 隨堂測驗 36](#_Toc497772129)

[2.7 存取修飾字(Access modifier) 37](#_Toc497772130)

[2.7.1 重點摘要 37](#_Toc497772131)

[2.7.2 隨堂測驗 38](#_Toc497772132)

[2.8 多型(Polymorphism) 38](#_Toc497772133)

[2.8.1 重點摘要 38](#_Toc497772134)

[2.8.2 隨堂測驗 39](#_Toc497772135)

[2.9 抽象類別(Abstract class) 40](#_Toc497772136)

[2.9.1 重點摘要 40](#_Toc497772137)

[2.9.2 隨堂測驗 40](#_Toc497772138)

[2.10 介面(Interface) 41](#_Toc497772139)

[2.10.1 重點摘要 41](#_Toc497772140)

[2.10.2 隨堂測驗 41](#_Toc497772141)

[3 軟體測試與除錯 42](#_Toc497772142)

[3.1 軟體測試方式與偵錯技巧 43](#_Toc497772143)

[3.1.1 重點摘要 43](#_Toc497772144)

[3.1.2 隨堂測驗 45](#_Toc497772145)

[3.2 軟體開發流程、軟體測試流程 46](#_Toc497772146)

[3.2.1 重點摘要 46](#_Toc497772147)

[3.2.2 隨堂測驗 48](#_Toc497772148)

# 程式語言基礎

## Java資料型別(Data type)

### 基本資料型別(Primitive data type)

* 整數: byte short int long char
* 浮點數: float double
* 布林: boolean

#### 隨堂測驗

|  |
| --- |
| (1)  下列何者是Java的基本資料型別？  (A) integer  (B) long  (C) bit  (D) decimal  答案: B |

### 參考資料型別(Reference data type) /類別(Class)

## 運算子(Operator)

### 常用運算子

* 一元運算子： ++ -- +(正) -(負) ! (type)
* 算術運算子： + - \* / %
* 關係運算子： > >= < <= == !=
* 邏輯運算子： && ||
* 條件運算子： ? :
* 指定/算術指定運算子： = += -= \*= /= %=

### 運算子優先順序

1. 一元運算子
   * v1++ v1--
   * ++v1 --v1 + - ! (type)
2. 算術運算子
   * \* / %
   * + -
3. 關係運算子
   * > >= < <=
   * == !=
4. 邏輯運算子
   * &&
   * ||
5. 條件運算子： ? :
6. 指定/算術指定運算子：= \*= /= += -= %=

### 隨堂測驗

|  |
| --- |
| (1)  程式語言一般支援算術、關係、邏輯等多種運算子（Operators），而這些運算子會具有優先順序。下列有關於Java或Objective-C運算子的優先順序的排序(由高到低)，何者是正確的？   1. + == && 2. && == + 3. == + && 4. + && ==   答案: A |

|  |
| --- |
| (2)  下列程式片斷執行後之輸出何者正確？  int num = 10;  System.out.print( num>5?num--:--num );  (A) 10  (B) 11  (C) 12  (D) 9  答案: A |

|  |
| --- |
| (3)  下列何者是Java提供的運算子？  (A) #  (B) @  (C) ~  (D) %  答案: D |

|  |
| --- |
| (4)  下列Java程式片斷執行後v2之值為何？  int v1=10;  int v2 = --v1-v1;  (A) v2之值為0  (B) v2之值為9  (C) v2之值為-20  (D) v2之值為-1  答案: A |

|  |
| --- |
| (5)  下列程式片斷執行後，s1之值為何？  **int** v1 = 100 / (**double**)3;  String s1 = String.*format*("%.2f", v1);  (A) "33.33"  (B) "33.333333"  (C) "33"  (D) 編譯有誤(Type mismatch: cannot convert from double to int)  答案: D |

|  |
| --- |
| (6)  程式語言支援運算子 (Operators)搭配運算元 (Operands)組成運算式(Expressions）以執行相關運算作業。下列Java或Objective-C程式片段執行後之結果何者為正確？  int v1 = 100;  int v2 = v1++;  (A) v1之值為101 且 v2之值為101  (B) v1之值為100 且 v2之值為100  (C) v1之值為101 且 v2之值為100  (D) v1之值為100 且 v2之值為101  答案: C |

## 控制結構(Control structure)

### 重複結構(Repetition structure)

#### 重點摘要

* for 迴圈

|  |
| --- |
| for (initialization; condition; increment) {  statement(s);  }  **int** total = 0;  **for**(**int** i=1;i<=10;i++){  total += i;  } |

* while 迴圈

|  |
| --- |
| while ( condition ) {  statement(s);  }  **int** total = 0, i = 1;  **while**(i<=10){  total += i;  i++;  } |

* do-while 迴圈

|  |
| --- |
| do {  statement(s);  }while( condition );  **int** total = 0, i = 1;  **do**{  total += i;  i++;  }**while**(i<=10); |

#### 隨堂測驗

|  |
| --- |
| (1)  Java提供那些迴圈敘述？  (A) for敘述  (B) while敘述  (C) do-while敘述  (D) repeat敘述  答案: A,B,C |

|  |
| --- |
| (2)  下列程式片斷執行後，j之值為何？  int j = 0;  for(int i=1; i<=10; i++){  j++;  if(i>6) break;  }  (A) 5  (B) 6  (C) 7  (D) 8  答案: C |

|  |
| --- |
| (3)  以下程式執行結果，何者正確？  int total = 0,i,j;  for(i=0;i<3;i++)  for(j=0;j<i;j++)  total += j;   1. total之值為1 2. total之值為2 3. total之值為3 4. total之值為4   答案: A |

|  |
| --- |
| (3)  public static void main(String[] args) {  int i = 0;  for (int j = 0; j < 4; j++)  i += 2;  System.out.println(i);  }  請問執行結果為何？  (A) 0  (B) 4  (C) 6  (D) 8  答案: D |

|  |
| --- |
| (4)  int[] x= {2,4,18};  int r=0;  for(int i=0;i<=x.length;i++)  r+=x[i];  請問如果在螢幕上輸出r，則會在螢幕上看到什麼？  (A) 6  (B) 3  (C) 24  (D) 錯誤訊息  答案: D |

### 選擇結構(Selection structure)

#### 重點摘要

* if 敘述

|  |
| --- |
| if ( condition ) {  statement(s);  }  **int** v1 = 100;  **if**(v1%2==0)  System.***out***.println("偶數");  執行結果：  I/System.out: 偶數 |

* if-else 敘述

|  |
| --- |
| if ( condition ) {  statement(s);  }  else{ 註：else前面一定要有if與之配對  statement(s);  }  int v1 = 101;  if(v1%2==0)  System.*out*.println("偶數");  else  System.*out*.println("奇數");  執行結果：  I/System.out: 奇數 |

* switch 敘述

|  |
| --- |
| switch(expression) {  case value1:  statement(s);  break;  case value2:  statement(s);  break;  default:  statement(s);  }  **int** v1 = 10;  **switch**(v1%2){  **case** 0:  System.***out***.println("偶數");  **break**;  **case** 1:  System.***out***.println("奇數");  **break**;  }  執行結果：  I/System.out: 偶數 |

#### 隨堂測驗

|  |
| --- |
| (1)  public static void main(String[] args) {  int a = 4, b = 0;  switch (a) {  case 2:  b += 2;  case 4:  b += 3;  default:  b += 5;  case 0:  b += 4;  break;  case 6:  b += 1;  }  System.out.println("b = " + b);  }  請問執行這段程式之後，b的值為多少？  (A) b = 13  (B) b = 3  (C) b = 8  (D) b = 12  答案: D |

|  |
| --- |
| (2)  為了提高程式維護性，下列哪一種流程控制不建議使用？  (A) if else  (B) goto  (C) switch case  (D) block  答案: B |

|  |
| --- |
| (3)  public static void main(String[] args) {  int count = 2;  if(count > 0)  System.out.println("A");  System.out.println("B");  else  System.out.println("C");  }  　請問執行結果為何？   1. ABC 2. AB 3. C 4. Compilation fails（編譯錯誤）   答案: D |

|  |
| --- |
| (4)  程式語言一般支援選擇性控制結構。下列有關於Java或Objective-C程式語言之選擇性敘述（Selection Statements），何者正確？   1. if後面一定要有else才能組成合法敘述 2. if後面可有多個else以組成合法敘述 3. else前面一定要有if才能組成合法敘述 4. 以上皆是   答案: C |

## 變數有效範圍(Scope)

### 主要分類

#### 實體變數(Instance variable)

* 在類別內宣告的非靜態的變數

#### 類別變數(Class variable)

* 在類別內宣告的靜態變數

#### 區域變數(Local variable)

* 在類別的方法內宣告的變數
* 有效範圍僅限於宣告該變數的方法內
* 在方法中宣告的輸入參數(Parameters)也屬於區域變數
* 同一個方法內所宣告的區域變數名稱不可重複
* 區域變數在方法被呼叫時才會被配置記憶體空間

#### 區塊變數(Block variable)

* 在區塊內宣告的變數
* 內層區塊可以使用外層區塊宣告的變數; 反之則否

**public** **static** **void** main(String[] args) {

**int** v1 = 100;

{

**~~int~~** ~~v1;~~ //Duplicate local variable v1

v1=200;

**int** v2 = 300;

}

~~v2 = 400;~~ //v2 cannot be resolved to a variable

}

### 隨堂測驗

|  |
| --- |
| (1)  程式語言會使用「變數」來暫存資料，其中「區域變數」（Local Variables）具有特定限制與特性，有關於Java的「區域變數」，下列敘述何者不正確？  (A) 在程式的方法（Methods）內宣告的變數稱為區域變數  (B) 區域變數的有效範圍僅限於宣告該變數的方法內  (C) 在程式的方法定義中宣告的輸入參數（Parameters）也屬於該方法的區域變數  (D) 方法內宣告的區域變數可以和宣告的輸入參數同名  答案:D |

|  |
| --- |
| (2)  下列有關變數的描述何者是錯誤的？  (A) 靜態變數又稱為類別變數  (B) 靜態變數可稱為同類別物件共同使用的資料  (C) 靜態變數又稱為實體變數  (D) 靜態變數一般可直接透過宣告該變數的類別名稱來存取  答案: C |

|  |
| --- |
| (3)  下列有關物件導向程式語言的變數（Variables）的描述，何者不正確？   1. 實體變數（Instance Variables）可被物件內的所有方法使用 2. 區域變數（Local Variables）是在方法內宣告的變數 3. 在方法內之內層區塊可以使用外層區塊的區域變數 4. 區域變數不能和實體變數同名   答案: D |

## 識別名稱(Identifier)

### Java識別名稱的基本規則

* 開頭字元：Unicode之字母 \_ $
* 第二個(含)以後之字元：Unicode之字母 \_ $ 數字
* 區分大小寫
* 不可使用關鍵字(keyword)(如 int、char、default、switch等)

### 隨堂測驗

|  |
| --- |
| (1)  程式語言一般會要求變數宣告時必須指定變數的識別名稱（Identifier），下列何者不是Java合法的識別名稱？  (A) num3  (B) \_num3  (C) $num3  (D) 3num  答案: D |

|  |
| --- |
| (2)  下列何者是Java的關鍵字？  (A) default  (B) var  (C) constant  (D) loop  答案: A |

## 函式/方法(Function/Method)

### 重點摘要

* 程式中使用方法之原因
  + 應用個別擊破法(divide-and-conquer approach)，降低程式複雜度
  + 提高軟體可重用性(software reusabitity)
  + 避免重複程式碼
* Java方法兩要素
  1. 定義(definition)
  2. 呼叫(method call)

//1.方法定義

**class** Utility{

**public** **static** **double** sum(**double** a, **double** b){

**return** a+b;

}

**public** **static** **double** sum(**double**[] nums){

**double** total = 0.0;

**for**( **double** num : nums)

total += num;

**return** total;

}

}

//2.方法呼叫(call by value)

Utility.sum(100, 200); //回傳 300.0

**double**[] nums = {1.0,2.0,3.0};

Utility.*sum*(nums); //回傳 6.0

### 隨堂測驗

|  |
| --- |
| (1)  程式語言支援函式（Functions）或方法（Methods）。關於使用函式或方法的優點，下列敘述何者不正確？  (A) 簡化程式複雜度  (B) 避免程式碼重複  (C) 增加程式可重用性  (D) 加快程式執行速度  答案: D |

|  |
| --- |
| (2)  下列有關方法的描述何者是錯誤的？  (A) 方法相當於模擬現實世界之物件的行為  (B) 一個方法只可以宣告一個參數  (C) 方法可分為類別方法與實體方法  (D) 靜態方法又稱為類別方法  答案: B |

## 例外處理(Exception handling)

### 重點摘要

* 例外處理(exception handling)機制讓Java程式在執行中可以捕捉並處理錯誤，以避免程式產生錯誤結果
* 例外處理器(exception handler)乃負責恢復程式錯誤之程式段
* 例外處理器組成結構

try{

statements;

}

catch(ExceptionType1 e){ }

catch(ExceptionType2 e){ }

finally{ }

* + catch區塊與finally區塊兩者可只存在一個

**try**{

**double** n = 100/0;

}

**catch**(ArithmeticException e){

System.***out***.println(e.getMessage());//**/ by zero**

}

**catch**(Exception e){

//statement(s)

}

**finally**{

//statement(s)

}

### 隨堂測驗

|  |
| --- |
| (1)  關於Java語言的例外處理（Exception Handling）機制，下列敘述何者不正確？  (A)例外處理是一種針對程式執行時所產生的錯誤所採取的處理措施  (B) Java語言一般會支援使用try-catch區塊來實作例外處理  (C)例外處理包括Error、Exception、RuntimeException等類別  (D) Java語言並沒有支援try-catch-finally例外處理區塊  答案: D |

|  |
| --- |
| (2)  在應用程式開發上，當程式碼執行時發生錯誤，會拋出例外資訊（Exception），於程式開發上，該如何捕捉例外資訊（Exception）並處理？  (A) if...else...  (B) switch  (C) try...catch...finally  (D) log.d(...);  答案: C |

## 逃逸序列(Escape sequence)

### 常用逃逸序列

* \n：換行(Line Feed)
* \t：水平定位(Horizontal Tab)
* \r：游標回歸當行開始處(Carriage Return)
* \\：反斜線
* \"：雙引號

### 隨堂測驗

|  |
| --- |
| (1)  Java或Objective-C程式語言事先定義一組「逃逸序列」(Escape Sequences)以代表各種特定字元。下列何者不是Java或Objective-C的「逃逸序列」？  (A) \n  (B) \t  (C) \c  (D) \r  答案: C |

## 註解(Comment)

### Java提供之註解

* // (Inline comment)
* /\* \*/ (Block comment)

### 隨堂測驗

|  |
| --- |
| (1)  撰寫程式時應該善用「註解」（Comments）以增進程式的可讀性。下列何者是Java或Objective-C的常用註解符號？   1. \\ 2. # 3. /\* \*/ 4. --   答案: C |

# 物件導向程式語言

## 物件(Object)導論

### 重點摘要

* 真實世界的物件都具有兩項特性：
  + **狀態**(State)或**屬性**(Attribute)
  + **行為**(Behavior)
* 物件導向程式設計(Object-Oriented Programming)以軟體的對應項模仿真實的世界; 它把**變數**(對應上述物件之屬性)與**方法**(對應上述物件之行為)整合成一個單元，稱為類別(class)，類別經實體化後即成物件。
* 類別相當於藍圖(Blueprint)，依藍圖建構起來的實體就是物件。

**new** Cat(); //將類別實體化(即建立物件)

//定義類別

**class** Cat{

//State|Attribute

String name = "Cat";

**int** age = 1;

//Behavior

**void** eat(){

//statement(s)

}

}

### 隨堂測驗

|  |
| --- |
| (1)  下列有關物件導向程式設計的描述何者是錯誤的？  (A) 真實世界的物件具有兩項特性: 狀態與行為  (B) 類別是設計圖  (C) 物件是設計圖  (D) 類別實體化就是物件  答案: C |

|  |
| --- |
| (2)  物件導向的特性，下列選項何者不正確？   1. 封裝（Encapsulation） 2. 多型（Polymorphism） 3. 繼承（Inheritance） 4. 抽象（Abstract）   答案: D |

## 繼承(Inheritance)

### 重點摘要

* + 物件導向程式設計允許一個類別繼承其他類別的變數(屬性)與方法(行為)，這種機制稱為繼承。
  + 繼承是一種軟體重複利用的方法
  + 繼承表達一種關係：子類別 是一種("is a") 父類別的東西，例如蘋果是水果的一種(Apple is a Fruit.)
  + 父類別(Superclass)又稱基礎類別(Base class); 子類別(Subclass)又稱為衍生類別(Derived class)或擴充類別(Extended class)
  + 繼承關係圖具有階層式之樹狀結構，因此子類別又可分成**直接子類別**與**間接子類別**兩種
  + Java只支援單一繼承，不支援多重繼承

**class** Fruit **extends** Object{

}

**class** Apple **extends** Fruit{

}

* + 繼承不是has-a的關係：has-a代表一個物件擁有另外一個物件的參考。例如下式 A dog has a mouth.

**class** Dog{

Mouth m = **new** Mouth( );

}

**class** Mouth{ }

### 隨堂測驗

|  |
| --- |
| (1)  物件導向的主要特徵為何？  (A) 繼承、封裝、多載  (B) 繼承、封裝、重構  (C) 繼承、多載、覆寫  (D) 繼承、封裝、多型  答案: D |

|  |
| --- |
| (2)  下列有關Java的父類別與子類別的描述何者是正確的？  (A)子類別又可分成直接子類別與間接子類別  (B)子類別只會繼承父類別的變數  (C) 子類別只會繼承父類別的方法  (D)以上皆是  答案: A |

|  |
| --- |
| (3)  下列何者是子類別與父類別之間的關係？  (A) has-a  (B) is-a  (C) get-a  (D)以上皆非  答案: B |

|  |
| --- |
| (4)  下列選項何者呈現物件導向中的關係：”A是一個B且擁有C”？  (A) class A extends C { private B b; }  (B) class B implements A { private C c; }  (C) class A extends B { private C c; }  (D) class C implements A { private B b; }  答案: C |

|  |
| --- |
| (5)  Java或Objective-C物件導向程式語言支援使用下列那一個關鍵字，使子類別可以呼叫父類別中的方法？  (A) over  (B) super  (C) parent  (D) above  答案: B |

|  |
| --- |
| (6)  「繼承」是物件導向程式語言的重要概念之一。下列有關「繼承」的描述，何者不正確？   1. 繼承可分成單一繼承或多重繼承兩種 2. Java或Objective-C程式語言僅支援多重繼承 3. Java或Objective-C程式語言支援多階層式的繼承關係 4. 子類別又稱為衍生類別（Derived Class）   答案: B |

|  |
| --- |
| (7)  public class A {  public int a = 123;  }  public class B extends A {  public int a = 456;  }  如果執行下列程式碼：  B x = new B();  System.out.println( String.valueOf(x.a));  則螢幕上會看到下列何者？  (A) 123  (B) 456  (C) 123,456  (D) 456,123  答案: B |

## 封裝(Encapsulation)

### 重點摘要

* 封裝是一種將類別內的變數私有化(private)而提供公有(public)方法讓其他類別存取這些私有變數的技術

**class** Time{

**private** **int** hour;

**public** **void** setHour(**int** hour){

**this**.hour = hour;

}

**public** **int** getHour(){

**return** hour;

}

}

### 隨堂測驗

|  |
| --- |
| (1)  關於Java語言的資料封裝（Data Encapsulation）機制，下列敘述何者正確？  (A) 一般利用物件的「private成員變數」搭配「private成員函式」而達成  (B) 一般利用物件的「public成員變數」搭配「private成員函式」而達成  (C) 一般利用物件的「private成員變數」搭配「public成員函式」而達成  (D) 一般利用物件的「public成員變數」搭配「public成員函式」而達成  答案: C |

|  |
| --- |
| (2)  你正在設計一個類別 (Class)，因為某些因素你將一部分的欄位規劃成公有成員 (public），另外將一部分的欄位規劃成私有成員 (private)，請問這樣的行為稱之為？  (A) 繼承 (Inheritance)  (B) 封裝 (Encapsulation)  (C) 介面 (Interface)  (D) 多型 (Polymorphism)  答案: B |

## 建構子(Constructor)

### 重點摘要

* 每當類別被實體化為物件時，該類別之建構子即被自動呼叫
* 建構子的特性
  + 建構子是物件內的特殊方法(Methods)
  + 與類別同名
  + 不可指定回傳值型別或是回傳一個值
  + 一個類別可能含有多個**多載建構子**(即建構子名稱相同，但參數列不同者)
  + 當類別中未定義任何建構子時，編譯器會自動為該類別產生一個不具參數的建構子，該建構子稱為**預設建構子(default constructor)**
  + 程式設計師**原則上**必須在類別建構子主體中之第一行，利用**super()/super(arguments)**敘述，呼叫父類別之建構子(無此敘述者，編譯器自動代為呼叫super(); )
  + 建構子通常用來初始化實體變數

**Snoopy sn = new Snoopy("小奴");**

**sn.name //小奴**

**class** Dog {

**protected** String name;

**public** Dog(){

//statement(s)

}

**public** Dog(String name){

**this**.name = name;

//statement(s)

}

}

**class** Snoopy **extends** Dog{

**public** Snoopy(){

**super**();

//statement(s)

}

**public** Snoopy(String name){

**super**(name);

//statement(s)

}

}

### 隨堂測驗

|  |
| --- |
| (1)  下列有關建構子(Constructors)的描述何者是正確的？  (A) 建構子一般用來初始化區域變數  (B) 建構子一般用來初始化實體變數  (C) 建構子不可多載  (D) 建構子是一種特殊的變數  答案: B |

|  |
| --- |
| (2)  假設C類別繼承B類別且B類別繼承A類別，當C類別實體化之後，各類別建構子完成執行的先後順序(由先到後)為何？  (A) C🡪B🡪A  (B) A🡪B🡪C  (C) A  (D) C  答案: B |

|  |
| --- |
| (3)  *6*  物件導向程式語言一般支援多階層繼承機制，下列有關類別的建構子(Constructors)的執行順序，何者正確？   1. 最下層的子類別的建構子最先完成執行 2. 最上層的父類別的建構子最先完成執行 3. 中間層的類別的建構子最先完成執行 4. 執行順序不固定   答案: B |

## 覆寫(Override)

### 重點摘要

* 覆寫是一種子類別**重新定義**從父類別繼承而來的方法的機制
* 子類別覆寫方法的基本原則
  + **方法名稱**與父類別之方法**相同**
  + **參數列**與父類別之方法的參數列**相同**
  + **回傳值之資料型別**與父類別之方法的回傳值之資料型別**相同或是子類別**
  + **存取修飾字**的可見度不可降低

**new** Snoopy().run(); //回傳 "5 m/s"

**class** Dog {

**public** String run(){

**return** "10 m/s";

}

}

**class** Snoopy **extends** Dog{

**public String run(){**

**return "5 m/s";**

**}**

}

### 隨堂測驗

|  |
| --- |
| (1)  物件導向程式語言支援方法覆寫 (Override)機制，下列描述何者正確？  (A) 方法覆寫即子類別重新定義父類別的建構子  (B) 方法覆寫就是一種方法多載 (Method Overloading)  (C) 子類別欲覆寫父類別的方法時，除了少數例外，原則上方法名稱、 參數列、回傳資料型別應該相同  (D) 以上皆是  答案: C |

|  |
| --- |
| (2)  *7*  有關覆寫（Override）的限制與注意事項，下列敘述何者不正確？   1. 存取權限須小於原方法 2. 方法中的參數列，不論數量、資料型別及擺放順序都必須相同 3. 覆寫是發生在有繼承關係的類別中 4. 若方法有回傳值，其回傳值型態需相同或原方法回傳值型別的子類別   答案: A |

|  |
| --- |
| (3)  public class A{  protect int methodA(int x){return 0;}  }  class B extends A{  //在此插入程式碼  }  下列程式碼各別插入於class B中，共有幾行可正確編譯？  public int methodA(int x){return 1;}  private int methodA(int x){return 0;}  protected int methodA(int x, int y){return 1;}  public String methodA(String x){return "A";}   1. 1 2. 2 3. 3 4. 4   答案: C |

## 方法多載(Method overloading)

### 重點摘要

* 方法多載是指同一個類別可以定義具有相同**方法名稱**但不同**參數**(指**參數個數**不同或**對應參數之資料型別**不同)的多個方法的機制

**class** Utility{

**public** **static** **double** sum(**double** a, **double** b){

**return** a+b;

}

**public** **static** **double** sum(**double**[] nums){

**double** total = 0.0;

**for**( **double** num : nums)

total += num;

**return** total;

}

}

### 隨堂測驗

|  |
| --- |
| (1)  下列有關方法多載的描述何者是錯誤的？  (A) 多載方法的方法名稱必須相同  (B) 多載方法的對應參數的資料型別與個數不可以完全相同  (C) 多載方法的回傳資料型別可以不同  (D) 多載方法的參數名稱必須不同  答案: D |

|  |
| --- |
| (2)  關於下列虛擬碼，挑選出最適合的答案  class A() {  void func1() { }  }  class B 繼承 A {  void func1() { ... }  void func1(int i) { ... }  void func1(String s, int i){ ... }  }  class C 繼承 A {  void func1() { ...}  }  void main() {  A x = new B();  A y = new C();  }  請問類別B中的func1(int i),func1(String s, int i)屬於下列哪一項行為？   1. 複寫（Overriding） 2. 繼承（Inheritance） 3. 多載（Overloading） 4. 多型（Polymorphism）   答案: C |

## 存取修飾字(Access modifier)

### 重點摘要

* 存取修飾字用來控制一個類別內的變數與方法可否被其他類別存取
* 存取修飾字依可見度(Visibility)由大到小排列如下:
  + **public**

一個類別的public變數與方法可被**所有類別**存取

* + **protected**

一個類別的protected變數與方法只可被**相同套件**(package)**的類別**或**所有子類別(包含不同套件之子類別)**存取

* + **no modifier** (package-private)

一個類別的package-private變數與方法只可被**相同套件**(package)**的類別**存取

* + **private**

一個類別的private變數與方法只可被**自己類別**存取

**public** **int** v1;

**protected** **int** v2;

**int** v3;

**private** **int** v4;

### 隨堂測驗

|  |
| --- |
| (1)  Java語言一般支援實體變數存取控制機制，下列敘述何者正確？  (A) protected實體變數可以被任何類別定義的方法直接存取  (B) private實體變數可以被任何子類別定義的方法直接存取  (C) public實體變數可以被任何子類別定義的方法直接存取  (D) 以上皆是  答案: C |

|  |
| --- |
| (2)  關於封裝 (Encapsulation)的種類，下列敘述何者為誤？  (A) private:只有class自己本身可存取，對應修飾字為private  (B) default:與class相同套件 (package)皆可存取，無對應修飾字  (C) protected:只有class自己本身以及子類別可存取，對應修飾字為 protected  (D) public:所有class皆可存取，對應修飾字為public  答案: C |

## 多型(Polymorphism)

### 重點摘要

* 多型是Java讓一個物件具有多種形狀(Forms)的機制
* 用**父類別**型別之指標指向子類別型別之物件時，該指標原則上無法呼叫子類別之方法，除非子類別方法**覆寫**父類別方法，此機制即為多型。

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Animal a1;

a1 = **new** Dog();

a1.printName(); //Dog

a1 = **new** Cat();

a1.printName(); //Cat

}

}

**class** Animal{

**public** **void** printName(){

System.***out***.println("Animal");

}

}

**class** Dog **extends** Animal{

**public** **void** printName(){

System.***out***.println("Dog");

}

}

**class** Cat **extends** Animal{

**public** **void** printName(){

System.***out***.println("Cat");

}

}

### 隨堂測驗

|  |
| --- |
| (1)  Java語言支援「多型」（Polymorphism）機制。關於「多型」，下列敘述何者不正確？  (A) 「多型」的字面意思為「具有多種形態」  (B) 「多型」一般發生於父類別型別的指標指向子類別物件的情況下  (C) 「多型」發生的關鍵之一在於「方法多載（Method Overloading）」  (D) 「多型」讓透過父類別物件參考呼叫子類別物件的方法成為可行  答案: C |

## 抽象類別(Abstract class)

### 重點摘要

* 具有抽象方法之類別稱為抽象類別
* 抽象方法指具有方法簽章(signature)而未實作之方法
* 抽象類別無法被實體化
* 抽象類別主要功能在提供一個父類別，讓其子類別能夠繼承它的方法並實作之

**abstract** **class** Shape{

**public** **abstract** String getName();

}

**class** Square **extends** Shape{

**public** String getName(){

**return** "Square";

}

}

### 隨堂測驗

|  |
| --- |
| (1)  下列有關抽象類別的描述何者是錯誤的？  (A) 抽象類別是具有抽象方法的類別  (B) 抽象方法是指具有方法簽章(signature)而無實作之方法  (C) 抽象類別主要功能在充當一個父類別  (D) 以上皆非  答案: D |

## 介面(Interface)

### 重點摘要

* 介面定義以interface關鍵字開始，基本上可包含之項目計有：
  + public abstract 方法
  + public final static 變數
  + public default/static 方法 (Java 8)
* 一個類別若實作一個介面，就如同跟編譯器簽定協定：我將會實作介面中定義之方法。 原則上該類別須實作介面內所定義之所有抽象方法，否則該類別即變成抽象類別
* Java支援單一繼承; 但支援實作多個介面

**interface** Shape{

**public** **abstract** String getName();

**public** **static** **final** **double** ***PI*** = 3.14;

**public** **default** **void** method1(){ }

}

**class** Circle **implements** Shape{

**public** String getName(){

**return** "Circle";

}

### 隨堂測驗

|  |
| --- |
| (1)  下列有關介面(Interface)的描述何者是錯誤的？  (A) 介面內可含有public abstract方法  (B) 每個類別只可以實作一個介面  (C) 介面內可含有public static final變數  (D) 介面擴充了Java單一繼承的限制，讓類別可以具有多種型別的能力  答案: B |

|  |
| --- |
| (2)  請問下列哪一項，是用來要求一個類別（Class）必須實作特定的方法？  (A) 繼承（Inheritance）  (B) 封裝（Encapsulation）  (C) 介面（Interface）  (D) 多型（Polymorphism）  答案: C |

|  |
| --- |
| (3)  關於Java語言中的Interface，下列敘述何者不正確？  (A) Interface不能包含有實作程式碼  (B) Interface必須透過class實作程式碼  (C) Interface的方法不能被覆載（override）  (D) Interface可以被new關鍵字實體化，但必須有匿名類別實作  答案: C |

# 軟體測試與除錯

## 軟體測試方式與偵錯技巧

### 重點摘要

(參考資料來源: 軟體工程/李允中教授/高立圖書有限公司)

* 廣義**軟體測試**，包括軟體品質的分析與檢驗，即所謂的**驗證與確認**(Verification and Validation, V&V)。
* 軟體測試工作需要事先做完整且詳細的**規劃**，而規劃的產出為**測試計畫書**。
* 驗證與確認需執行**靜態分析**與**動態測試**。
* **靜態分析**
  + **不直接執行軟體系統**，而是以人工或自動化方法**評估**軟體開發各階段的**工作產品**，檢查是否滿足所制訂的需求規格，並嘗試找出錯誤與及早修正。
  + 靜態分析執行的方法包含
  1. **檢視(Inspection)**
* 是一種最正規結構化(Formal Structure)的團隊審查程序。
  1. **結構化逐步審查(Structured Walkthrough)**
* 是一種輕量化檢視(Inspection-Lite)方法，要求必須有計畫、有結構的進行審查，但此檢視方法不正式與不嚴謹。
  1. **主動審查(Active Review)**
  + 業界常用的**同儕審查(Peer Review)**就是一種**靜態分析**方法
* **動態測試**
  + 程式撰寫完成後才能執行軟體程式碼，進行動態測試。
  + 動態測試需事先設計**測試案例**(Test Case)，然後執行測試案例以及收集、評估結果資料。
  + 設計測試案例一般使用**黑箱(Black Box)測試**與**白箱(White Box)測試**兩種技術。
    - **白箱測試**是一種根據**程式碼**產生測試案例與測試資料的技術，成本比較高。
    - **黑箱測試**目的在檢驗輸入正確資料時是否會產生正確結果，成本比較低。
  + 白箱測試的主要技術
    - **基本路徑測試(Basic Path Testing)**：
      * 最基本的測試方式，藉由找出所有的獨立路徑，用以確認程式碼中**每一行敘述**都被測試案例執行過，可以發現簡單的程式敘述撰寫錯誤。
    - **邏輯條件測試(Logic Condition Testing)**：
      * 透過檢測程式碼中每個**邏輯條件**的組合，來設計可能的測試案例資料。
    - **資料流測試(Data Flow Testing)**：
      * 主要是從程式碼中資料**變數**的使用上去測試測試案例資料。
    - **迴圈測試(Looping Testing)**：
      * 主要是針對程式碼片段中若有**迴圈**的設計時，如何設計測試案例資料，以正確的測試迴圈的敘述，發現可能的問題。
  + 黑箱測試的主要技術
    - **等價分類法(Equivalence Partition)**：
      * 將**輸入資料**分成幾個等級(值域)，然後分別加以測試。
    - **邊界值分析法(Boundary Value Analysis)**：
      * 查看在極大值或極小值周圍的執行情形是否有錯，因為許多錯誤常發生在輸入或輸出資料條件的邊界值。
      * 本方法為等價分類法的擴展，邊界值通常是等價類別的界限
    - **因果圖(Cause-Effect Graph)**：
      * 當需求規格有比較複雜的邏輯說明時，適合使用此種方法。

### 隨堂測驗

|  |
| --- |
| (1)  同儕審查（Peer Review）是一種軟體測試靜態分析方法。下列有關同儕審查常用的審查方式，何者不正確？  (A) 檢視（Inspection）  (B) 結構化逐步審查（Structured Walkthrough）  (C) 主動審查（Active Review）  (D) 被動審查（Passive Review）  答案: D |

|  |
| --- |
| (2)  軟體測試規畫應產出下列哪項產品？  (A) 測試計畫書  (B) 測試報告書  (C) 程式設計規格書  (D) 以上皆是  答案: A |

|  |
| --- |
| (3)  黑箱測試為設計測試案例的主要技術之一，下列何者不是黑箱測試的優點？  (A)測試案例設計的成本高  (B)不受測試案例多寡的影響皆能獲得良好的測試效果  (C) 容易測到程式內部細節  (D) 以上皆是  答案: D |

|  |
| --- |
| (4)  下列那一項不是白箱測試常用的測試案例設計方法？  (A)等價分類法(Equivalence Partition)  (B)基本路徑測試(Basic Path Testing)  (C)迴圈測試(Looping Testing)  (D)資料流測試(Data Flow Testing)  答案: A |

|  |
| --- |
| (5)  有關軟體測試中的靜態分析與動態分析，下列敘述何者正確？   1. 通常會先進行動態分析後才會進行靜態分析 2. 靜態分析會直接執行軟體，觀察其結果 3. 動態分析不直接執行軟體，而是以人工或自動化方式評估各階段的產品是否有達到需求規格 4. 動態分析中有黑箱測試、白箱測試等技術來測試案例   答案: D |

## 軟體開發流程、軟體測試流程

### 重點摘要

* 軟體開發過程中，透過下列各種**軟體測試階段/策略**，來檢查軟體是否符合使用者需求與選擇性地確保軟體執行的正確性。
  + 單元測試(Unit Testing)
    - 由**軟體開發人員**親自對軟體內部結構執行最小規模的測試。
    - 通常使用**白箱測試**方法來設計測試案例
  + 整合測試(Integration Testing)
    - 測試模組或子系統的介面整合是否正常運作
    - 主要由**軟體測試人員**進行整合測試
    - 通常使用**黑箱測試**方法來設計測試案例
  + 系統測試(System Testing)
    - 檢驗系統中各子系統之間的整合，與整個系統軟硬體功能及執行績效的需求是否符合
    - 主要由**硬體工程師與軟體測試人員**進行系統測試
    - 通常使用**黑箱測試**方法來設計測試案例
  + 驗收測試(Acceptance Testing)
    - 測試系統的整體性運作是否符合使用者的需求
    - 主要由**軟體系統操作者**來進行測試，並使用實際完成的軟體系統與實際操作數據來進行測試
    - 通常使用**黑箱測試**方法來設計測試案例
    - 驗收測試根據其**測試的方式**與**測試環境**的不同，又可再細分以下兩種測試：
      * **阿法(α)測試**：

由軟體開發人員指導使用者在**系統開發所在地**進行操作，開發者可同時記錄錯誤和使用時所產生的問題。

* + - * **貝塔(β)測試**：

在**終端用戶處**由使用者自行進行測試 (開發者不在現場)，使用者須自行記錄問題並回復給開發者。

* 軟體測試程序中的任一**軟體測試階段**，一般都可以分成以下四個主要的**執行步驟**
  + 測試規劃：測試前應擬定測試計畫及測試目標
  + 測試準備：準備所有參考資料及測試工具與環境
  + 執行軟體測試：執行實際測試與記錄錯誤與缺失
  + 評估與控制：監控與檢討測試成效
* 所有測試階段都可實施效能測試(Performance Testing)，一般可分為：
  + 負載測試(Load Testing)
    - 主要在測試電腦、周邊設備、伺服器以及網路的運作狀況，藉以找出元件的**限制、瓶頸或錯誤**。
  + 壓力測試(Stress Testing)
    - 主要是測試電腦、網路、程式或設備**在不適當的狀態**下仍具有一定的效用。

### 隨堂測驗

|  |
| --- |
| (1)  在測試軟體時也是有階段性的，請問下列何者為正確的軟體測試階段？  (A) 單元測試→整合測試→系統測試→驗收測試  (B) 單元測試→系統測試→整合測試→驗收測試  (C) 系統測試→單元測試→整合測試→驗收測試  (D) 系統測試→整合測試→單元測試→驗收測試  答案:A |

|  |
| --- |
| (2)  有關軟體動態測試策略與測試案例設計方法的描述，下列何者錯誤？  (A) 整合測試通常使用黑箱測試方法設計測試案列  (B) 系統測試通常使用黑箱測試方法設計測試案列  (C) 驗收測試通常使用白箱測試方法設計測試案列  (D) 單元測試通常使用白箱測試方法設計測試案列  答案: C |

|  |
| --- |
| (3)  有關單元測試的相關描述，下列何者正確？  (A) 單元測試是由軟體開發人員自己親自執行的測試  (B) 單元測試主要使用黑箱測試方法來設計測試案例  (C) 單元測試屬於軟體動態測試策略中最高層次的測試  (D) 以上皆是  答案: A |

|  |
| --- |
| (4)  整合測試設計需考量的要點，下列選項何者不正確？   1. 需考慮系統執行而需要的測試環境及測試環境與生產環境的差別 2. 測試案例之執行策略 3. 測試案例運行時需要的外部條件 4. 由開發工程師進行   答案: D |

|  |
| --- |
| (4)  關於系統測試的階段說明，下列描述何者有誤？  (A) 單元測試：各獨立單元模組在與系統其他模組隔離的情況下進行 測試，檢查每個程式模組是否實現了規定的功能  (B) 整合測試：是在單元測試的基礎上將已經通過測試的單元模組按 照設計要求組裝成系統或子系統進行測試的活動  (C) 系統測試：透過整合測試的軟體，同其運作環境、資料和使用者 結合在一起，在實際或模擬實際環境下，對系統進行全面的測試  (D) 驗收測試：以開發者為主的測試，由開發者設計測試案例，使用 實際資料進行測試  答案: D |

|  |
| --- |
| (5)  請問下列哪一項測試主要是測試電腦、網路、程式或設備在負荷較重的狀態下，仍保有一定的效用？  (A) 負載測試  (B) 壓力測試  (C) 集中測試  (D) 煙霧測試  答案:B |

|  |
| --- |
| (6)  下列何者是軟體測試階段的主要執行步驟之一？  (A) 測試規劃  (B) 測試準備  (C) 測試執行  (D) 以上皆是  答案:D |

|  |
| --- |
| (7)  驗收測試依據測試方式與環境的不同，又可再分成兩種測試，下列敘述何者正確？   1. 阿法（α）測試由使用者在自己的工作場所進行 2. 貝塔（β）測試由軟體開發人員陪同使用者在系統開發場所進行 3. 貝塔（β）測試由使用者在自己的工作場所進行 4. 以上皆非   答案:C |

|  |
| --- |
| (8)  下列何者為程式元件在通過單元測試後應該開始進行的測試，以找出系統整合時的錯誤？  (A) 系統測試（System Testing）  (B) 整合測試（Integration Testing）  (C) 效能測試（Performance Testing）  (D) 黑箱測試（Black Box Testing）  答案:B |