TQC+ 物件導向程式語言 Java 認證

簡志峰

認證介紹 - 基礎物件導向程式語言

- 本認證為操作題，總分為100分。
- 操作題為第一至五類各考一題共五大題，第一大題至第五大題每題20分，總計100分。
- 於認證時間60分鐘內作答完畢，成績加總達70分（含）以上者該科合格。

等級	考科代碼	測驗時間	學科	術科
專業級	SJP3	60分鐘	無	5題

Techficiency Quotient Certification TQC+®

Java™

認證介紹 - 物件導向程式語言

- 本認證皆為操作題，總分為100分。
- 第一類第至六類各考一題共六大題，第一大題至第五大題每題10分、第六大題50分，總計100分。
- 於認證時間100分鐘內作答完畢，成績加總達70分（含）以上者該科合格。

等級	考科代碼	測驗時間	學科	術科
專業級	PJP3	100分鐘	無	6題

Techficiency Quotient Certification TQC+®

Java

題型規劃 - 物件導向程式語言

- 第一類到第五類是單元題，每類出一題，每題 10分。
- 第六類是題組題，裡面有5小題，前後是有連貫性的，50分。

題型規劃

- 第一類：基本認識
- 第二類：條件判斷式
- 第三類：迴圈
- 第四類：遞迴程式設計
- 第五類：陣列設計能力
- 第六類：物件導向程式設計與例外處理

應考建議

1. 拿到試卷時先看整份題目。前面未必容易，後面未必困難。看完整份題目後排列先後順序再進行。
2. 100 分鐘，6大類，扣除檔案開啟、儲存時間，前五大類請在一半的時間內完成(40分鐘)，時間自行掌握。
3. 程式要能執行，不能執行就沒有分數。
4. 以通過(70分合格)為目標，因此請務必掌握大部份題目(第一~五類要掌握，第六類至少前三題能掌握)。
5. 請了解/熟悉流程. 考試時以平常心應考！

加強題目

1. 第一類 : 102 110
2. 第二類 : 202 204
3. 第三類 : 304 306
4. 第四類 : 402 404
5. 第五類 : 504 506
6. 第六類 : 602 610

題型規劃

- 第一類：基本認識
- 第二類：條件判斷式
- 第三類：迴圈
- 第四類：遞迴程式設計
- 第五類：陣列設計能力
- 第六類：物件導向程式設計與例外處理

102 - 單位換算

- 畫面顯示[Please input:]，之後由鍵盤輸入數字後，輸出轉換數值(由公斤轉換成磅數，單位轉換:1公斤等於2.20462磅)，執行結果如以下範例。

執行結果

```
Please input: 15
15.000000 kg = 33.069300 ponds
```

```
import java.util.Scanner;

public class JPA102 {

  public static void main(String[] args) {

    System.out.println("Please input:");

    // 使用Scanner這個方法來讀取鍵盤輸入

   double k = new Scanner(System.in).nextDouble();

   System.out.println(k + " kg = " + (k * 2.20462) + "
ponds");

  }

}
```

110 - 圓形面積

- **請撰寫三個方法計算下面圖形的面積，並輸出總面積。**
- **圓的半徑=5，PI=3.1415926，圓面積計算公式:半徑平方*PI，請寫出calCircle函數計算圓面積。**
- **三角形的底=10，高=5，三角形面積公式:底*高/2，請寫出calTriangle函數計算三角形面積。**
- **長方形的長=5，寬=10，長方形面積公式:長*寬，請寫出calRectangle函數計算長方形面積。**
- **圖形面積=圓面積+三角形面積+長方形面積，執行結果如範例圖。**

執行結果

```
圓形面積為：78.539815
三角形面積為：25.000000
方形面積為：50.000000
此圖形面積為：153.539815
```

110 - 圓形面積

```
public class JPA110 {
public static void main(String args[]) {
double totalarea;
System.out.printf("圓形面積為：%f \n", calCircle(5));
System.out.printf("三角形面積為：%f \n", calTriangle(10, 5));
System.out.printf("方形面積為：%f \n", calRectangle(10, 5));
totalarea = calCircle(5) + calTriangle(10, 5) + calRectangle(10, 5);
System.out.printf("此圖形面積為：%f \n", totalarea);
}
// 宣告一個計算圓面積的方法
public static double calCircle(int a) {
return (a * a * 3.1415926);
}
// 宣告一個計算三角形面積的方法
public static double calTriangle(int a, int b) {
// 記得要將a*b的乘積強制轉型成double，這樣除出來數字才會保留小數部分
return ((double) (a * b) / 2);
}
// 宣告一個計算長方形面積的方法
public static double calRectangle(int a, int b) {
return ((double)(a * b));
}
}
```

題型規劃

- 第一類：基本認識
- 第二類：條件判斷式
- 第三類：迴圈
- 第四類：遞迴程式設計
- 第五類：陣列設計能力
- 第六類：物件導向程式設計與例外處理

202 - 比較大小

- 畫面顯示**[Input:]**，並於下方要求輸入兩個整數，中間以空格分開。
- 依輸入的兩個整數比較大小，重複二次，執行結果如以下範例。

執行結果

```
Input:
50 88
88 is larger than 50
Input:
33 45
45 is larger than 33
```

```
import java.util.*;
public class JPA202 {
  public static void main(String[] args) {
    input();
    input();
  }
public static void input() {
  System.out.println("Input:");
  Scanner sc = new Scanner(System.in);
  int a, b;
  a = sc.nextInt();
  b = sc.nextInt();
// 結果可能有三種狀況，a>b，a=b，a<b，所以建立條件判斷式處理如下
  if (a > b)
    System.out.printf("%d is larger than %d\n", a, b);
  else if (b > a)
    System.out.printf("%d is larger than %d\n", b, a);
  else
    System.out.printf("%d is equal to %d\n", a, b);
  }
}
```

204 - 公倍數計算

- 畫面顯示**[Input:]**，並於下方要求輸入一個整數。

- 計算整數是否能同時被**5**、**9**整除，若是則印出**[Yes]**，否則印出**[No]**，重複二次，執行結果下。

執行結果

```
Input:
90
Yes
Input:
70
No
```

```
import java.util.*;

public class JPA204 {
    public static void main(String[] args) {
    input();
    input();
  }
public static void input() {
  System.out.println("Input:");
  Scanner sc = new Scanner(System.in);
  int a = sc.nextInt();
  // 使用and的運算，並且同時符合mod5=0和mod9=0的數字進行判斷
  if ((a % 5) == 0 && (a % 9) == 0)
    System.out.printf("Yes\n");
  else
    System.out.printf("No\n");
  }
}
```

題型規劃

- 第一類：基本認識
- 第二類：條件判斷式
- 第三類：迴圈
- 第四類：遞迴程式設計
- 第五類：陣列設計能力
- 第六類：物件導向程式設計與例外處理

304 - 餐點費用

- 畫面顯示**[Please enter meal dollars -1 too stop:]**，並於後方要求輸入一個點餐費用。
- 重複執行直到輸入**-1**為止，輸入**-1**後輸出點餐數量、所有點餐的總費用及平均負擔費用，平均值取自小數第二位，執行結果如以下範例。

執行結果

```
Please enter meal dollars or enter -1 to stop: 180
Please enter meal dollars or enter -1 to stop: 120
Please enter meal dollars or enter -1 to stop: 99
Please enter meal dollars or enter -1 to stop: 399
Please enter meal dollars or enter -1 to stop: 150
Please enter meal dollars or enter -1 to stop: -1
餐點總費用:948
5 道餐點平均費用為: 189.60
```

304 - 餐點費用

```
import java.util.*;
public class JPA304 {
  public static void main(String[] args) {
    double total = 0;
    int input = 0;
    int count = 0;
    double average;
    // 先在迴圈外面要求使用這輸入
    System.out.print("Please enter meal dollars or enter -1 to stop: ");
    input = new Scanner(System.in).nextInt();
    // 進入迴圈時會檢查，使否符合條件
    while (input != -1) {
    // 進入迴圈後，第一件事便是進行運算
      count++;
      total = total + input;
    // 運算完後再次要求使用者輸入
      System.out.print("Please enter meal dollars or enter -1 to stop: ");
      input = new Scanner(System.in).nextInt();
    }
    average = total / count;
    System.out.println("餐點總費用:" + total);
    System.out.printf(" %d 道餐點平均費用為: %.2f %n", count, average);
  }
}
```

306 - 迴圈次方計算

- 畫面顯示**[Inout:]**，並於下方要求輸入二個整數 **m**、**n**，並以空格鍵分隔。
- 並用一個類別方法及 **while loop** 計算 **m** 的 **n** 次方，直到輸入**m=999**為止，執行結果如範例圖。

執行結果

```
Input:
2 2
4
Input:
100 7
100000000000000
Input:
999
```

306 - 迴圈次方計算

```
import java.util.Scanner;
public class JPA03 {
  static Scanner keyboard = new Scanner(System.in);
  public static void main(String argv[]) {
    int m = 0, n;
    while (m != 999) {
      System.out.println("Input:");
      m = keyboard.nextInt();
      if (m != 999) {
        n = keyboard.nextInt();
        System.out.println(powerOf(m, n));
      }
    }
  }
 static long powerOf(int m, int n) {
    long result = 1;
    while (n >= 1) {
      result *= m;
      n--;
    }
    return result;
  }
}
```

題型規劃

- 第一類：基本認識
- 第二類：條件判斷式
- 第三類：迴圈
- 第四類：遞迴程式設計
- 第五類：陣列設計能力
- 第六類：物件導向程式設計與例外處理

402 - 尾端遞迴階乘計算

- **畫面顯示[Input n (0 <= n <= 16):]，並於後方要求輸入一個整數。**

- **分別使用尾端遞迴及迴圈計算 n 的階乘，直到輸入為999為止，執行結果如以下範例。**

執行結果

```
Input n (0 <= n <= 16):6
6 的階乘(尾端遞迴) = 720
6 的階乘(迴圈) = 720
Input n (0 <= n <= 16):8
8 的階乘(尾端遞迴) = 40320
8 的階乘(迴圈) = 40320
Input n (0 <= n <= 16):999
```

402 - 尾端遞迴階乘計算

```
import java.util.*;
public class JPA402 {
  public static void main(String[] args) {
    int size = 0;
    do {
      System.out.print("Input n ( 0 <= n <= 16 ):");
      size = new Scanner(System.in).nextInt();
    } while (size > 16 && size != 999 || size < 0 && size != 999);

    while (size != 999) {
      // 迴圈方法
      int L = facLoop(size);
      System.out.printf("%d的階乘(迴圈) = %d\n", size, L);
      // 尾端遞迴
      int T = facRec(size, 1);
      System.out.printf("%d的階乘(尾遞迴) = %d\n", size, T);
      do {
        System.out.print("Input n ( 0 <= n <= 16 ):");
        size = new Scanner(System.in).nextInt();
      } while (size > 16 && size != 999 || size < 0 && size != 999);
    }
  }
```

```
// 迴圈方法:使用for-loop
public static int facLoop(int a) {
  int sum = 1;
  for (int b = 1; b <= a; b++)
    sum = b * sum;
  return sum;
}

// 尾端遞迴:不斷呼叫自己的方法，進行運算
public static int facRec(int a, int b) {
  if (a == 1)
    return b;
  else
    return facRec(a - 1, a * b);
  }
}
```

404 - 遞迴最大公因數

- **畫面顯示[Input m:]，並於後方要求輸入一個整數m。**
- **畫面顯示[Input n:]，並於後方要求輸入一個整數n。**
- **使用尾端遞迴計算 m 與 n 的最大公因數，直到輸入為999為止，執行結果如以下範例。**

執行結果

```
Input m: 7
Input n: 49
最大公因數為: 7
Input m: 64
Input n: 128
最大公因數為: 64
Input m: 15
Input n: 10
最大公因數為: 5
Input m: 999
```

404 - 遞迴最大公因數

```
import java.util.*;
public class JPA404 {
  public static void main(String args[]) {
    System.out.print("Input m :");                          // 先請使用者輸入m值
    Scanner sc = new Scanner(System.in);
    int m = sc.nextInt();
    while (m != 999) {                                      // 迴圈判斷，m要不等於999才可進入
    System.out.print("Input n :");                          // 進入迴圈後再請使用者輸入n值
      int n = sc.nextInt();                                 // 進行最大公因數計算
      System.out.println("最大公因數為:"+R(m, n));          // 迴圈最後在要求使用者輸入下一個迴圈需要的m，若此m在進入下一個迴圈時條件不過，則不進入迴圈了
      System.out.print("Input m :");
      m = sc.nextInt();
    }
  }
static int R(int m, int n) {                                // gcd 求最大公因數演算法
    if (m % n == 0)                                         // 如果mod為零時，則找到最大公因數，回傳值
      return n;
    else                                                    // 如果mod數不為零時，則把mod得到的數寫入m，再次進行gcd
    {
      int tem = n;
      n = m % n;
      m = tem;
      return R(m, n);
    }
  }
}
```

題型規劃

- 第一類：基本認識
- 第二類：條件判斷式
- 第三類：迴圈
- 第四類：遞迴程式設計
- 第五類：陣列設計能力
- 第六類：物件導向程式設計與例外處理

504 - 費氏數

- 請先宣告一個大小為10的陣列，並將最前面二個陣列指定為費氏數的初始值，再用初始使值計算

  其餘的費氏數，再以分析方式，顯示此費氏數的前10個數值，執行結果如以下範例。

  執行結果

```
0
1
1
2
3
5
8
13
21
34
```

504 - 費氏數

```
public class JPA504 {
  public static void main(String[] args) {
    int[] n = new int[10];
    // 初始化前兩個數
    n[0] = 0;
    n[1] = 1;
    // 費式數列前十個
    for (int a = 2; a < 10; a++)
      n[a] = n[a - 1] + n[a - 2]; // 陣列的index由2開始，其目的在於相加前面兩個index=1 and 0
    for (int a = 0; a < 10; a++)
      System.out.println(n[a]);
   }
}
```

506 - 三維陣列元素之和

- **計算陣列 A 內所有元素的總和，執行結果如範例圖。**

```
int A[4][2][3] =
    {{{1,2,3},{4,5,6}},
    {{7,8,9},{10,11,12}},
    {{13,14,15},{16,17,18}},
    {{19,20,21},{22,23,24}}};
```

執行結果

```
sum = 300
```

506 - 三維陣列元素之和

```
public class JPA05 {
    public static void main(String[] argv) {
        int sum =0;
        //觀察此三維陣列是由四個二維，二個一維，一維裡面各三個元素組成
        int A[][][] = {{{1,2,3},{4,5,6}}, {{7,8,9},{10,11,12}}, {{13,14,15},{16,17,18}}, {{19,20,21},{22,23,24}}};
        //透過三個for-loop迴圈來相加
        for(int a=0;a<4;a++)
            for(int b=0;b<2;b++)
                for(int c=0;c<3;c++)
                    sum+=A[a][b][c];
        System.out.printf("sum = %d\n", sum);
    }
}
```