第十章 例外處理

10-1 例外處理簡介

10-1-1 何謂例外處理

程式執行當中，遇到異常狀況讓程式無法繼續執行時，系統必須有適當的處置，此處理狀況稱之為『例外處理』(Exception)，較常見的異常狀況有：

(1) 『I/O 處理』時發生異常狀況：譬如開啟某依檔案，但該檔案並不存在，而讓系統無法繼續執行時，所應處理的例外處理。

(2) 『運算處理』時發生異常狀況：譬如除數分母為 0 時。

(3) 『陣列處理』時發生異常狀況：陣列索引值超過陣列大小範圍，或索引值為負值時。

(4) 『資料形態處理』時發生異常狀況：譬如須由鍵盤讀入整數，但輸入為非數字時。

在 java 規範裡異常狀態有分為『例外』(Exception)與『錯誤』(Error)，一般來講例外是表示異常狀態較輕微的現象，而錯誤則表示較嚴重的現象。兩者皆宣告於 Java.lang 類別下，Exception 類別較常用的有：

Java.lang.Exception	說　　明
ArrayIndexOutOfBoundsExceptuin	陣列索引值超過範圍。
NegativeArraySizeException	陣列索引值為負值。
IllegalAccessException	資料型態存取不正確。
IOException	存取輸入/輸出裝置不正確。
NoSuchFieldException	載入類別檔案的欄位不正確。
NumberFormatException	數字格式不正確。
ArithmeticException	數學運算不正確。
RuntimeException	在 JVM 執行當中所發生的錯誤皆是。

另外，較常用的 Error 類別有：

Java.lang.Error	說　　明
ClassFormatError	導入的類別型態錯誤。
Error	所有可 Throwable 的錯誤。
IllegalAccessError	不合法的存取錯誤。
VirtualMachineError	虛擬機執行錯誤。
OutOfMemoryError	超出記憶體範圍錯誤。
NoSuchMethodError	缺少方法成員錯誤。

還有許多相關 Exception 與 Error 類別，請自行參閱 Java DOC。

10-1-2 例外處理的語法

Java 例外處理的語法如下：

```
try {
      // 監視可能發生異常現象的程式區塊
}
catch (例外類別_1 例外物件_1) {
      // 捕抓到例外事件_1，程式執行區塊
}
catch (例外類別_2 例外物件_2)
     // 捕抓到例外事件_2，程式執行區塊
}
….
finally {
     // 無論是否捕抓到例外事件，皆必須執行的城市區塊
}
```

(1) try 區塊：此區塊程式在執行當中，被監視是否發生例外事件。

(2) catch (事件類別_1 事件物件_1) 區塊：當 try 區塊內執行當中，發生例外事件_1 時，則被捕抓到，並執行此區塊內程式。

(3) final 區塊：無論是否捕抓到例外事件，此區塊都要執行。

10-1-3 範例研討：簡單捕抓異常現象

（A）程式範例：Ex10_1.java

我們用一個程式來驗證 NumberFormatException 與 ArrayIndexOutOfBoundsException 兩例外類別的運用，前者表示輸入數字資料格式不對發生例外事件，後者陣列指標超過範圍，程式範例如下：

```
01  //Ex10_1.java
02  import java.util.*;
03  public class Ex10_1 {
04      public static void main(String args[]) {
05          Scanner keyin = new Scanner(System.in);
06
07          double fract;    //分數;
08
09          double numer; //  分子
10
11          double demo; //  分母
12          int data[] = {1, 2, 3, 4, 5, 6, 7};
13          int point;
14          try {
15              System.out.printf("請輸入除數的分子  =>");
16
17              numer = keyin.nextDouble();
18
19              System.out.printf("請輸入除數的分母  =>");
20
21              demo = keyin.nextDouble();
22              fract = numer / demo;
23              System.out.printf("\n");
24              System.out.printf("%.2f/%.2f = %.2f\n", numer,demo,fract);
25
26              System.out.printf("請輸入幾個陣列元素輸出  =>");
27
28              point = keyin.nextInt();
29              for (int i=0; i<point; i++)
30                  System.out.printf("%d    ", data[i]);
31              System.out.printf("\n");
32          }
33          catch(InputMismatchException except1){
34              System.out.printf("Catch NumberFormatException\n");
35              System.out.printf("%s\n", except1.toString());
36          }
          catch(ArrayIndexOutOfBoundsException except2){
              System.out.printf("Catch ArrayIndexOutOfBoundsException\n");
```

```
37                  System.out.printf("%s\n", except2.toString());
38              }
39              finally {
40                  System.out.println("Program finally");
41              }
42          }
43  }
44
```

（B） 執行結果

(1) 正常現象，兩數輸入正確，輸出兩數箱除結果，陣列指標輸入在陣列元素內，但最後 finally 敘述皆會執行，如下：

```
D:\Java2_book\chap10>java Ex10_1
請輸入除數的分子 =>5
請輸入除數的分母 =>4
5.00/4.00 = 1.25
請輸入幾個陣列元素輸出 =>3
1   2   3
Program finally
```

(2) 當數字輸入不正確，而輸入字元時，發生補抓到 NumberFormatException 異常現象，如下：

```
D:\Java2_book\chap10>java Ex10_1
請輸入除數的分子 =>5
請輸入除數的分母 =>t
Catch NumberFormatException
java.util.InputMismatchException
Program finally
```

(3) 當陣列指標超過範圍時，則捕抓到 ArrayIndexOutOfBoundsException 異常現象，如下：

```
D:\Java2_book\chap10>java Ex10_1
請輸入除數的分子 =>3
請輸入除數的分母 =>1
```

```
3.00/1.00 = 3.00
請輸入幾個陣列元素輸出 =>8
1  2  3  4  5  6  7  Catch ArrayIndexOutOfBoundsException
java.lang.ArrayIndexOutOfBoundsException: 7
Program finally
```

10-2 擲出 throw 例外功能

10-2-1 利用 throw 擲出例外

程式執行當中，也可以拋出例外物件。但拋出的例外物件必須繼承 Throwable 類別或其衍生類別，Error 及 exception 類別就是。語法如下：

```
throw new 例外物件();
```

10-2-2 範例研討：除以零

（A）程式範例：Ex10_2.java

在 Java 內定中，如遇到除以零的現象，會告知產生無窮大的結果，不會產生例外現象，我們可以自行測試，讓他拋出 ArithmeticException 例外物件，程式範例如下：

```
01  //Ex10_2.java
02  import java.util.*;
03  public class Ex10_2 {
04      public static void main(String args[]) {
05          Scanner keyin = new Scanner(System.in);
06          double   a;   //分數;
07          double   b; //  分子
08          double   c; //  分母
09          try {
10              System.out.printf("請輸入除數的分子 =>");
11              b = keyin.nextDouble();
12              System.out.printf("請輸入除數的分母 =>");
13              c = keyin.nextDouble();
14              if (c == 0)
15                  throw new ArithmeticException();
16
17
18
19
```

```
                a = b / c;
                System.out.printf("\n");
                System.out.printf("%.2f/%.2f = %.2f\n", b,c,a);
            }
            catch(InputMismatchException except1){
                System.out.printf("Catch NumberFormatException\n");
                System.out.printf("%s\n", except1.toString());
            }
            catch(ArithmeticException except2){
                System.out.printf("Throw out Catch ArithmeticException\n");
                System.out.printf("%s\n", except2.toString());
            }
        }
}

```

（B）執行結果：

```
D:\Java2_book\chap10>java Ex10_2
請輸入除數的分子 =>5
請輸入除數的分母 =>0
Throw out Catch ArithmeticException
java.lang.ArithmeticException
```

10-2-3 範例研討：自行拋出例外

（A）程式範例：Ex10_3.java

如果對於輸入或運算結果並非期望值，也可利用 throw 拋出例外物件。下列程式期望輸入偶數，但輸入奇數時，讓他拋出例外事件，或輸入結果不正確，也拋出。範例如下：

```
//Ex10_3.java
import java.util.*;
public class Ex10_3 {
    public static void main(String args[]) {
        Scanner keyin = new Scanner(System.in);
        int k;
        try {
            System.out.printf("請輸入除數的分子 =>");
            k = keyin.nextInt();
```

```
11                  if (k <10 || k>20)
12                      throw new ArithmeticException();
13                  int x=20, y=2, m;
14
15                  System.out.printf("請輸入  %d/%d =>", x, y);
16                  m = keyin.nextInt();
17                  if (m != x/y)
18                      throw new Exception();
19              }
20              catch(ArithmeticException except1){
21                  System.out.printf("Catch NumberFormatException\n");
22                  System.out.printf("%s\n", except1.toString());
23
24                  System.out.printf("輸入並非介於  10  與  20  之間\n");
25              }
26              catch(Exception except2){
27                  System.out.printf("Catch Exception\n");
28                  System.out.printf("%s\n", except2.toString());
29              }
30          }
31  }
32
```

（B）執行結果：

```
D:\Java2_book\chap10>java Ex10_3    [正常輸入]
請輸入除數的分子  =>20
請輸入  20/2 =>10

D:\Java2_book\chap10>java Ex10_3    [輸入範圍不對拋出 NumberFormatException]
請輸入除數的分子  =>4
Catch NumberFormatException
java.lang.ArithmeticException
輸入並非介於  10  與  20  之間

D:\Java2_book\chap10>java Ex10_3       [輸入答案不對拋出 Exception]
請輸入除數的分子  =>20
請輸入  20/2 =>3
```

```
Catch Exception
java.lang.Exception
```

10-3 自訂擲出例外 - throws

10-3-1 throws 語法

吾人可將某一類別內方法所發生的例外事件，歸納於某一例外物件內，在管理方面也許比較容易，則在主方法內捕抓例外事件 (try {…} catch () { …})，其他方法則將事件擲出即可，不用再 try .. catch 命令擷取，如以下語法，說明如下：

(1) 在主方法內利用 try .. catch 擷取 Exception_1 例外事件。

(2) 方法 meth_1 與 meth_2 在 try 監視範圍內。

(3) 方法 meth_1 與 meth_2 在宣告語句內加入 **throws Exception_1**，表示有任何例外事件，則擲出到 Exception_1 物件內。

(4) 在方法監督事件發生就不需要 try … catch 捕抓，直接 throw new Exception_1 (" Message") 即可，其中 Message 表示顯示擲出訊息。

(5) 其中 Exception_1 例外物件必須是 Exception 與 Error 類別下物件。

```
public class UsingException {
    public static void main(String args[]) {
        …..;
        try {
            meth_1();
            ……;
            meth_2();
            ……;
        }
        catch ( Exception_1   except1) {
            System.out.printf("%s\n", except1.getMessage());
        }
    }
    public static void meth_1(..) throws Exception_1 {
        …;
        throw new Exception_1("Message_1");
```

```
            ….;
        }
        public static void meth_2(..) throws Exception_1 {
            …;
            throw new Exception_1(“Message_2”);
            ….;
        }
}
```

10-3-2 範例研討：throws ArithmeticException

（A）程式範例：Ex10_4.java

吾人用一個範例(Ex10_4.java)來說明自訂擲出例外物件。在主方法內捕抓ArithmeticException，其他方法則將例外事件丟到該物件內，程式範例如下：：

```
01  //Ex10_4.java
02  import java.util.*;
03  public class Ex10_4 {
04      public static void main(String args[]) {
05          Scanner keyin = new Scanner(System.in);
06          int k;
07          try {
08              System.out.printf("請輸入 10 ~ 20 之間的數 =>");
09              k = keyin.nextInt();
10              System.out.printf("輸入值為：%d\n", rangeTest(k)); // 執行方法成員
11              int x=20, y=2, m;
12              System.out.printf("請輸入 %d/%d =>", x, y);
13              m = keyin.nextInt();
14              if (divTest(x, y, m) ==1)                    // 執行方法成員
15                  System.out.printf("正確\n");
16          }
17          catch(ArithmeticException except1){
18              System.out.printf("Catch ArithmeticException\n");
19              System.out.printf("%s\n", except1.toString());
20              System.out.printf("%s\n", except1.getMessage());
21          }
22      }
23      public static int rangeTest(int k) throws ArithmeticException {     // 方法成員
```

```
29              if (k <10 || k>20)
30                      throw new ArithmeticException("輸入不在範圍內");
31          return k;
32      }
33      public static int divTest(int x, int y, int m) throws ArithmeticException{    //方法成員
34           if (m != x/y)
35                      throw new ArithmeticException("除的結果不對");
36          else
37                return 1;
38      }
39  }
40
41
42
43
```

（B）執行結果：

```
D:\Java2_book\chap10>java Ex10_4              [正常輸入]
請輸入 10 ~ 20 之間的數 =>12
輸入值為：12
請輸入 20/2 =>10
正確

D:\Java2_book\chap10>java Ex10_4     [輸入範圍不對拋出 ArithmeticException]
請輸入 10 ~ 20 之間的數 =>4
Catch ArithmeticException
java.lang.ArithmeticException: 輸入不在範圍內
輸入不在範圍內

D:\Java2_book\chap10>java Ex10_4     [除結果不對拋出 ArithmeticException]
請輸入 10 ~ 20 之間的數 =>13
輸入值為：13
請輸入 20/2 =>5
Catch ArithmeticException
```

```
java.lang.ArithmeticException: 除的結果不對
除的結果不對
```

10-3-3 範例研討：throws IOException

目前非常普遍利用 Java 來開發應用程式，對於輸入/輸出處理最為頭痛，時常會發生例外事件。尤其最常見的 App 或視窗的人機介面處理。Java 為了方便程式開發者，在 Java.io.IOException 套件裡宣告了許多例外事件，只要將它宣告擲入 (throws IOException)，就不需要利用 try {…} 區塊來捕抓例外事件產生。

(A) 程式範例：Ex10_5.java

此範例係利用 try {….} 區塊來捕抓例外事件發生。程式功能是利用串流方式由檔案 test.txt 讀取資料，再依序輸出到螢幕上，如果 test.txt 不存在，則發生例外事件。程式範例如下：

```
01 import java.io.*;
02
03 public class Ex10_5 {
04     public static void main (String[] args) {
05         FileInputStream fis = null;
06         int i=0;
07         char c;
08         try{
09             File file = new File("test.txt");
10             fis = new FileInputStream(file);
11             while ((i=fis.read()) != -1) {
12                 c = (char)i;
13                 System.out.print(c);
14             }
15         }catch (IOException e){
16             e.printStackTrace();
17         }
18     }
19
20 }
21
```

(B) 執行結果：

(1) 檔案不存在，則執行結果如下：

```
D:\Java2_book\chap10>java Ex10_5
java.io.FileNotFoundException: test.txt (系統找不到指定的檔案。)
        at java.io.FileInputStream.open0(Native Method)
        at java.io.FileInputStream.open(Unknown Source)
        at java.io.FileInputStream.<init>(Unknown Source)
        at Ex10_5.main(Ex10_5.java:10)
```

(2) 檔案存在，則執行結果如下：

```
D:\Java2_book\chap10>java Ex10_5
Good Luck To You
```

(C) 程式範例：Ex10_6.java

如果，我們在方法前面宣告 throws IOException，則不須利用 try {….} 區塊來捕抓例外事件發生，程式範例如下：

```
01 import java.io.*;
02
03 public class Ex10_6 {
04     public static void main (String[] args) throws IOException {
05         FileInputStream fis = null;
06         int i=0;
07         char c;
08         File file = new File("test.txt");
09         fis = new FileInputStream(file);
10         while ((i=fis.read()) != -1) {
11             c = (char)i;
12             System.out.print(c);
13         }
14     }
15
16 }
17
```

(D) 執行結果：

(1) 檔案不存在，則執行結果如下：

```
D:\Java2_book\chap10>java Ex10_6
Exception in thread "main" java.io.FileNotFoundException: test.txt (系統找不到指
定的檔案。)
        at java.io.FileInputStream.open0(Native Method)
        at java.io.FileInputStream.open(Unknown Source)
        at java.io.FileInputStream.<init>(Unknown Source)
        at Ex10_6.main(Ex10_6.java:10)
```

(2) 檔案存在，則執行結果如下：

```
D:\Java2_book\chap10>java Ex10_6
Good Luck To You
```