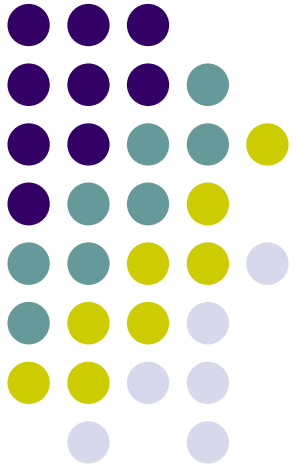
第十三章
例外處理

瞭解什麼是例外處理

認識例外類別的繼承架構

認識例外處理的機制

學習如何撰寫例外類別



例外的基本觀念

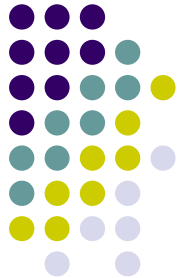
- 在撰寫程式時常見的幾種情況：

  (1) 要開啓的檔案並不存在

  (2) 存取陣列時，陣列的索引值超過陣列容許的範圍

  (3) 原本預期使用者由鍵盤輸入的是整數，但使用者輸入的卻是英文字母

  - 這類不尋常的狀況稱爲「例外」（exception）
  - 所有的例外都是以類別的型態存在



例外處理的優點

- 易於使用
- 可自行定義例外類別
- 允許我們拋出例外
- 不會拖慢執行速度
- 增進程式的穩定性及效率

簡單的例外範例

- app13_1是個錯誤的程式：

```
01  // app13_1, 索引值超出範圍
02  public class app13_1
03  {
04     public static void main(String args[])
05     {
06        int arr[]=new int[5];                  // 容許 5 個元素
07        arr[10]=7;                             // 索引值超出容許範圍
08        System.out.println("end of main() method !!");
09     }
10  }
```

預設例外處理機制會依下面的程序做處理：
(1) 拋出例外
(2) 停止程式執行

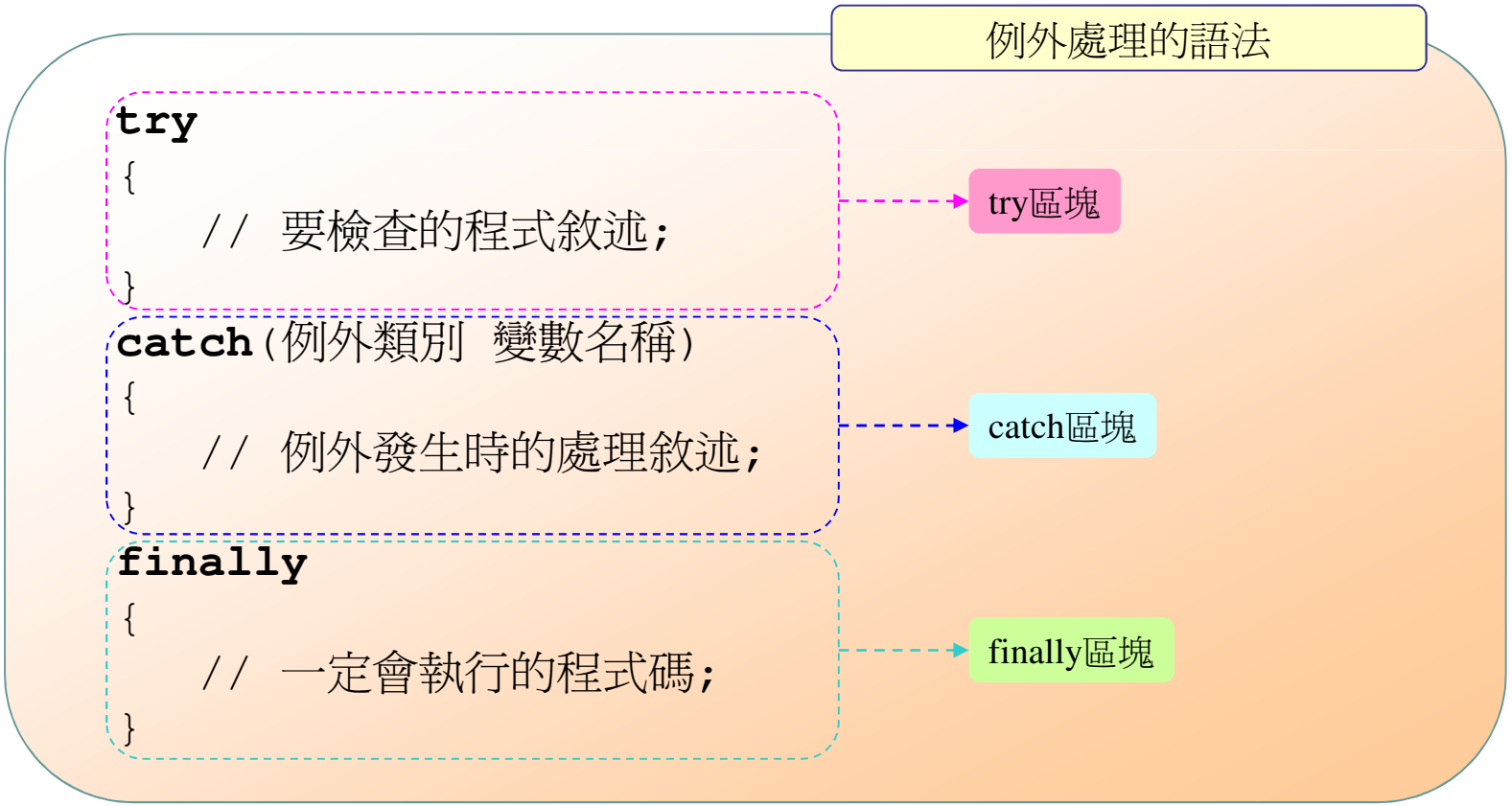
- 執行到第7行時，會產生下列的錯誤訊息：

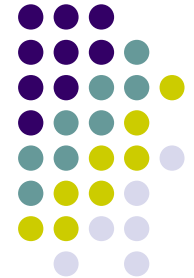
```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
    at app13_1.main(app13_1.java:7)
```

Array Index Out Of Bounds Exception，是 "陣列索引值超出範圍的例外" 之意

例外處理的語法

- 例外處理是由 try、catch與finally所組成的程式區塊，其語法如下：





例外處理的順序

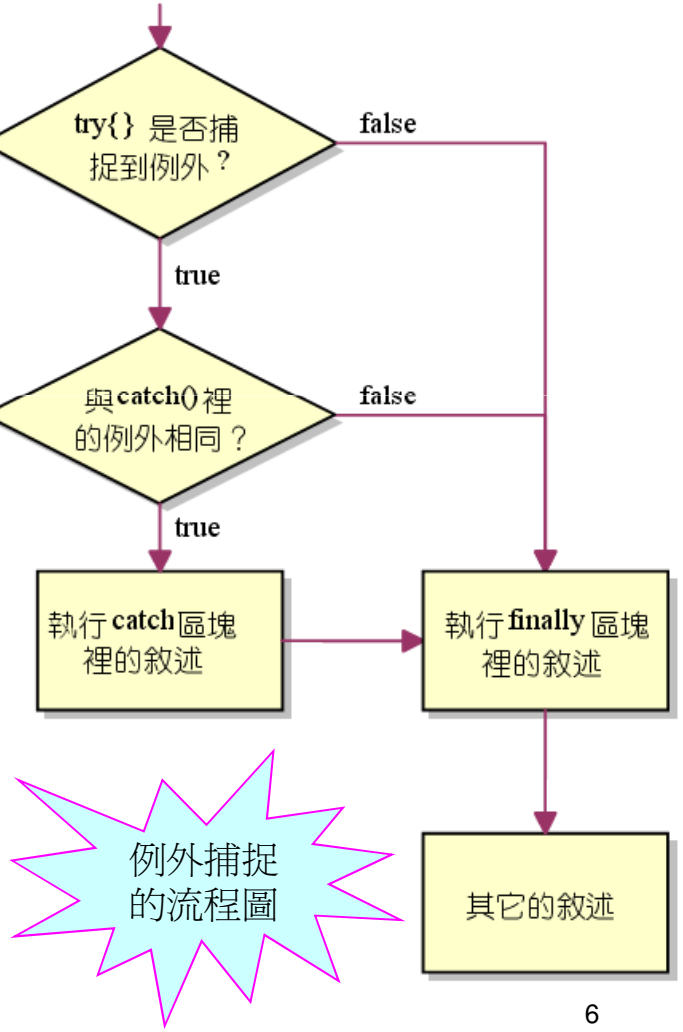
- 例外處理的順序：

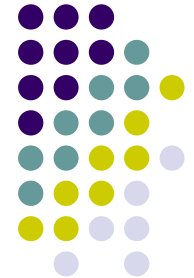
(1) try程式區塊若有例外發生時，程式的執行便中斷，並拋出"由例外類別所產生的物件"

(2) 拋出的物件如果屬於catch() 括號內欲捕捉的例外，則catch會捕捉此例外，然後進到catch的區塊裡繼續執行

(3) 無論try程式區塊是否有捕捉到例外，或者捕捉到的例外是否與catch() 括號裡的例外相同，最後一定會執行finally區塊裡的程式碼

(4) finally的區塊執行結束後，程式再回到try-catch-finally區塊後的地方繼續執行





例外處理的實例

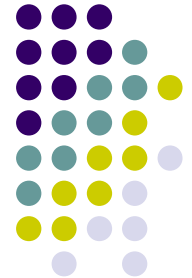
- app13_2是例外處理的範例：

```
01  // app13_2, 例外的處理
02   public class app13_2
03   {
04     public static void main(String args[])
05     {
06        try              // 檢查這個程式區塊的程式碼
07        {
08           int arr[]=new int[5];
09           arr[10]=7;
10        }
11        catch(ArrayIndexOutOfBoundsException e)
12        {
13           System.out.println("index out of bound!!");
14        }
15        finally          // 這個區塊的程式碼一定會執行
16        {
17           System.out.println("this line is always executed!!");
18        }
19        System.out.println("end of main() method!!");
20     }
21  }
```

如果拋例外，便執行此區塊的程式碼

```
/* app13_2 OUTPUT--------------

index out of bound!!
this line is always executed!!
end of main() method!!
-------------------------------*/
```



例外類別的變數

- 捕捉到例外時，例外類別會建立一個類別變數e，如：

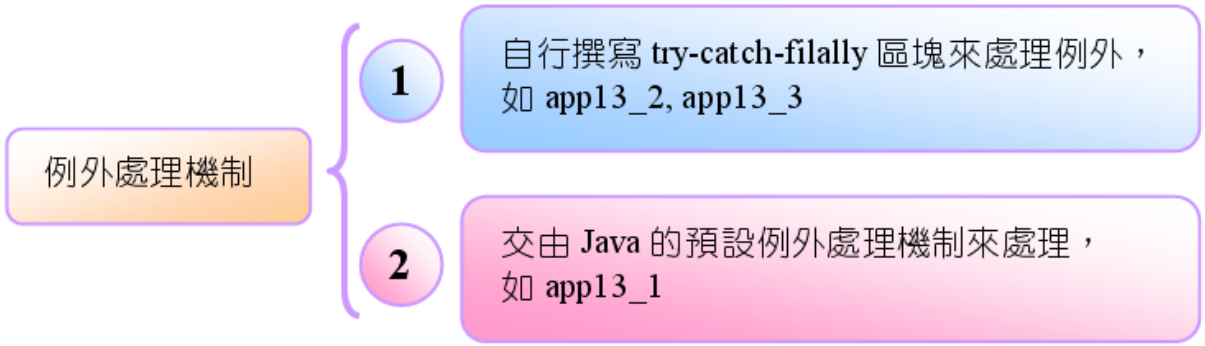
```
/* app13_3 OUTPUT-----------------------------------
index out of bound!!
Exception=java.lang.ArrayIndexOutOfBoundsException: 10
end of main() method !!
-------------------------------------------------*/
01  // app13_3, 例外訊息的擷取
02  public class app13_3
03  {
04     public static void main(String args[])
05     {
06        try
07        {
08           int arr[]=new int[5];
09           arr[10]=7;
10        }
11        catch(ArrayIndexOutOfBoundsException e)
12        {
13           System.out.println("index out of bound!!");
14           System.out.println("Exception="+e);      // 顯示例外訊息
15        }
16        System.out.println("end of main() method !!");
17     }
18  }
```

例外處理機制的回顧

- 例外發生時，通常有二種方法來處理
  - 一種是交由預設的例外處理機制做處理，如

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
    at app13_1.main(app13_1.java:7)
```

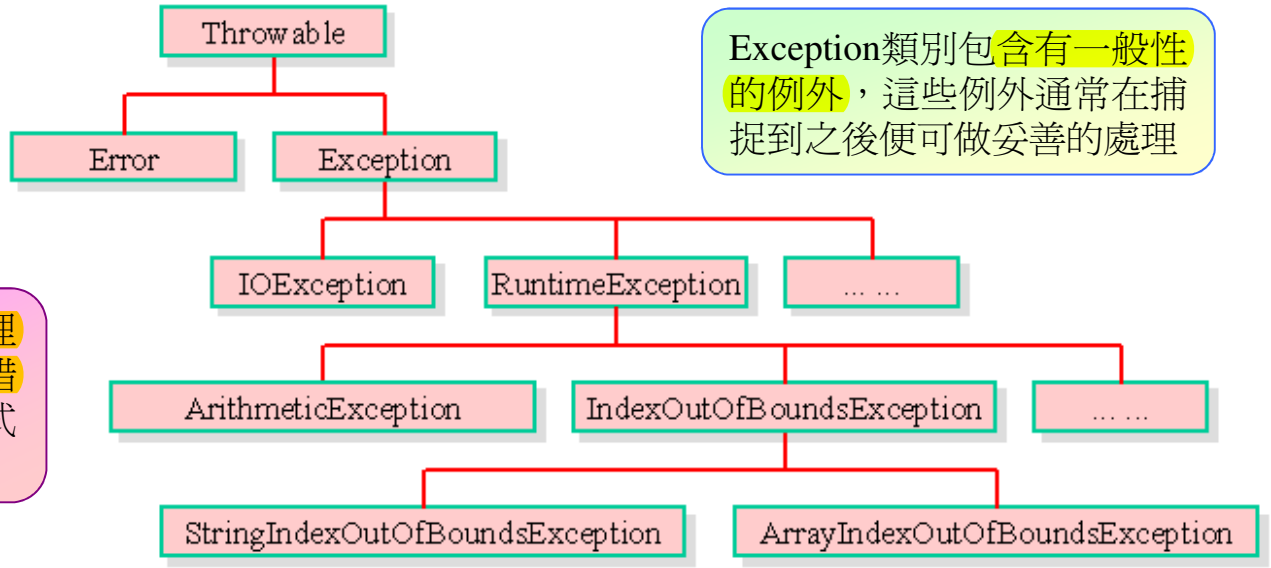
    接著結束程式的執行
  - 另一種方式是自行撰寫try-catch-finally區塊來捕捉例外
- 下圖繪出例外處理機制的選擇流程：



Throwable類別

- 例外類別可分爲兩大類：
  - java.lang.Exception類別
  - java.lang.Error類別
- 下圖爲Throwable類別的繼承關係圖

它們均繼承自java.lang.Throwable類別



Exception類別包含有一般性的例外，這些例外通常在捕捉到之後便可做妥善的處理

Error類別專門用來處理嚴重影響程式執行的錯誤，通常不會設計程式碼去捕捉這類的錯誤

catch() 括號的限制

- 例外發生時，catch() 只接收由Throwable類別的子類別所產生的物件

只接收由 Throwable 類別的子類別所產生的物件

```
catch( ArrayIndexOutOfBoundsException e )
{
   System.out.println("index out of bound!!");
   System.out.println("Exception="+e);       // 顯示例外訊息
}
```

捕捉例外

- 想捕捉**一種以上**的例外，就必須針對所有可能被拋出的例外撰寫catch() 程式碼，如：

```
01  try
02  {
03     // try 區塊的程式碼
04  }
05  catch(ArrayIndexOutOfBoundsException e)
06  {
07     // 捕捉到 ArrayIndexOutOfBoundsException 例外所執行的程式碼
08  }
09  catch(ArithmeticException e)
10  {
11     // 捕捉到 ArithmeticException 例外所執行的程式碼
12  }
```

- 想捕捉所有的例外，可以利用Exception例外，如：

```
01  catch(Exception e)
02  {
03     // 捕捉任何例外所執行的程式碼
04  }
```

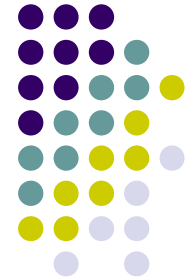
例外的拋出

- 拋出例外有下列兩種方式：
  (1) 於程式中拋出例外
  (2) 指定method拋出例外

- 於程式中拋出例外時的語法如下：

例外處理的語法

```
throw    由例外類別所產生的物件;
```



於程式中拋出例外

- app13_4是於程式中拋出例外的範例：

```
01  // app13_4, 於程式中拋出例外
02  public class app13_4
03  {
04     public static void main(String args[])
05     {
06        int a=4,b=0;
07
08        try
09        {
10          if(b==0)
11           throw new ArithmeticException();    // 拋出例外
12          else
13           System.out.println(a+"/"+b+"="+a/b); // 若沒有拋出例外，則執行此行
14        }
15        catch(ArithmeticException e)
16        {
17           System.out.println(e+" throwed");
18        }
19     }
20  }
```

```
/* app13_4 OUTPUT----------------------
java.lang.ArithmeticException throwed
-------------------------------------*/
```

throw關鍵字所接的是「由例外類別所產生的物件」，因此必須使用new關鍵字產生物件

系統自動拋出例外

- app13_5是讓系統自動拋出例外的驗證：

```
01  // app13_5, 讓系統自動拋出例外
02  public class app13_5
03  {
04     public static void main(String args[])
05     {
06        int a=4,b=0;
07
08        try
09        {
10           System.out.println(a+"/"+b+"="+a/b);
11        }
12        catch(ArithmeticException e)
13        {
14           System.out.println(e+" throwed ");
15        }
16     }        /* app13_5 OUTPUT-------------------------------------
17  }           java.lang.ArithmeticException throwed: / by zero throwed
              ----------------------------------------------------*/
```

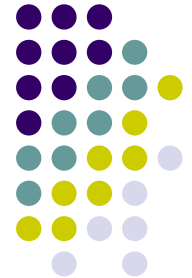
由method拋出例外

- 由method拋出例外的語法：

由method拋出例外

```
method名稱(引數...) throws 例外類別1，例外類別2,...
{
    // method內的程式碼
}
```

- 在method的內部拋出例外，是使用關鍵字「throw」
- 如果是指定要由method拋出例外，則使用「throws」



指定method拋出例外

- app13_6是指定由method來拋出例外的範例

```
01  // app13_6, 指定 method 拋出例外
02  public class app13_6
03  {
04     public static void aaa(int a,int b) throws ArithmeticException
05     {
06        int c;
07        c=a/b;
08        System.out.println(a+"/"+b+"="+c);
09     }
10
11     public static void main(String args[])
12     {
13        try                          /* app13_6 OUTPUT--------------------------------
14        {
15           aaa(4,0);                 java.lang.ArithmeticException: / by zero throwed
16        }                            ----------------------------------------------*/
17        catch(ArithmeticException e)
18        {
19           System.out.println(e+" throwed");
20        }
21     }
22  }
```

不同類別的method拋出例外

```
01  // app13_7, 從不同類別內的 method 拋出例外
02  class Ctest
03  {
04      public static void aaa(int a,int b) throws ArithmeticException
05      {
06         int c=a/b;
07         System.out.println(a+"/"+b+"="+c);
08      }
09  }
10
11  public class app13_7
12  {
13     public static void main(String args[])
14     {
15        try
16        {
17           Ctest.aaa(4,0);
18        }
19        catch(ArithmeticException e)
20        {
21           System.out.println(e+" throwed");
22        }
23     }
24  }
```

app13_7說明如何從不同類別裡的method裡拋出例外

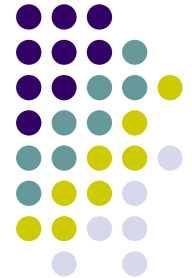
```
/* app13_7 OUTPUT--------------------------------
java.lang.ArithmeticException: / by zero throwed
-----------------------------------------------*/
```

自行撰寫例外

- 自己設計的例外類別必須繼承Exception類別
  - 自行撰寫例外類別的語法如下：

撰寫自訂例外類別的語法

```
class   例外類別名稱 extends Exception
{
    // 定義類別裡的各種成員
}
```

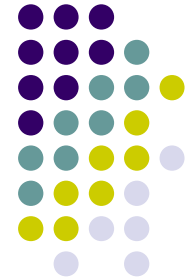


自行撰寫例外的範例 (1/2)

- 以一個範例來說明如何定義自己的例外類別：

```
01  // app13_8, 定義自己的例外類別
02  class CCircleException extends Exception    // 定義自己的例外類別
03  {
04  }
05
06  class CCircle          // 定義類別 CCircle
07  {                                                  由 method 拋出例外
08     private double radius;
09     public void setRadius(double r) throws CCircleException
10     {
11        if(r<0)
12        {
13           throw new CCircleException();          // 拋出例外
14        }
15        else
16           radius=r;
17     }
```

```
/* app13_8 OUTPUT---------
CCircleException throwed
area=0.0
-------------------------*/
```



自行撰寫例外的範例 (2/2)

```
18      public void show()
19      {
20         System.out.println("area="+3.14*radius*radius);
21      }
22   }
23
24   public class app13_8
25   {
26      public static void main(String args[])
27      {
28         CCircle cir=new CCircle();
29         try
30         {
31            cir.setRadius(-2.0);
32         }
33         catch(CCircleException e)    // 捕捉由 setRadius()拋出的例外
34         {
35            System.out.println(e+" throwed");
36         }
37         cir.show();
38      }
39   }
```

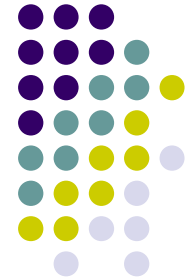
```
/* app13_8 OUTPUT---------
CCircleException throwed
area=0.0
-------------------------*/
```

IOException例外 (1/2)

- 會拋出IOException例外的method，其例外處理的方式有兩種：
  - 一種是直接由main() method拋出例外，讓預設的例外處理機制來處理，如：

```
01  // app3_13, 由鍵盤輸入字串
02  import java.io.*;
03  public class app3_13
04  {
05     public static void main(String args[]) throws IOException
06     {
..        ...
16     }
17  }
```

由 main()拋出例外讓系統預設的例外處理機制來處理



IOException例外 (2/2)

- 另一種方式是在程式碼內撰寫try-catch區塊來捕捉拋出的IOException例外，如

```
01  // app13_9, 撰寫 try-catch 區塊來捕捉 IOException 例外
02  import java.io.*;                    // 載入 java.io 類別庫裡的所有類別
03  public class app13_9
04  {
05     public static void main(String args[])
06     {
07        BufferedReader buf;
08        String str;
09
10        buf=new BufferedReader(new InputStreamReader(System.in));
11        try
12        {
13           System.out.print("Input a string: ");
14           str=buf.readLine();
15           System.out.println("string= "+str);          // 印出字串
16        }
17        catch(IOException e){}
18     }
19  }
```

```
/* app13_9 OUTPUT----------
Input a string: Hello Java!
string= Hello Java!
---------------------------*/
```