第十三章
例外處理(Exception)

本章學習目標
- ◆了解什麼是例外處理
- ◆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

13.1 例外的基本觀念

在執行程式時，經常發生一些不尋常的狀況(run time error)。例如：

(1) 要開啟的檔案不存在。
(2) 陣列的索引值超過了陣列容許的範圍。
(3) 使用者輸入錯誤。

Java把這類不尋常的狀況稱為「例外」（exception）。

所以，exception是在進行錯誤程式的資訊提供，使個人在處理錯誤時更方便。

程式錯誤的說明

- 程式的邏輯錯誤不易發現，若在程式中撰寫if-else來偵錯，無法捕捉所有錯誤，且效率極低。所以java中的exception可以便於拋出例外，且不會拖慢執行速度，增進程式的穩定性和效率。

- 錯誤的分類
  - 程式語法錯誤
    - ex char="SCJP"
  - Run time error
    - ex int 除以0
  - Login error
    - ex 利息計算公式錯誤

13.1.1為何需要例外處理？

13-3

- **例外處理**是現代程式語言處理程式執行過程中，可能發生問題（或者錯誤）時的一種較有效率的方法。

- 在過去，程式開發人員常花時間在撰寫「檢查錯誤」的相關程式碼，最主要原因是，當程式碼執行過程中，若發生「錯誤（Error）」時，將會中斷整個軟體的執行，造成程式無法繼續往下執行。

- **現代語言以「預先認定」某些程式片段（或執行某特定方法）可能出現 Exception例外，若事先因為認定其會發生例外，就要求在設計過程中一定要將處理例外情形的程式碼預先撰寫設計好，當執行過程中真的產生例外時，會按照事先設計的程式碼來處理例外，程式也能正常的繼續執行。**

程式中撰寫Exception的好處

- 易於使用
- 可以自行定義例外類別
- 允許拋出例外
- 不會拖慢執行速度

13-4

13.1.2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 error
8   System.out.println("end of main() method !!"); //此行不會被執行到
09     }
10  }
```

app13_1執行時，會產生下列的錯誤訊息

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

13-5

Java的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。

13-6

13.1.3 例外的處理

例外處理：加上捕捉例外的程式碼，可針對不同的例外做妥善的處理。

例外處理的語法如下：

```
try
  {
     // 將可能發生例外的地方
        用try包住;
  }
catch(例外類別  變數名稱)
  {
     // 例外發生時的處理敘述;
  }
finally
  {
     // 一定會執行的程式碼;
  }
```

try區塊

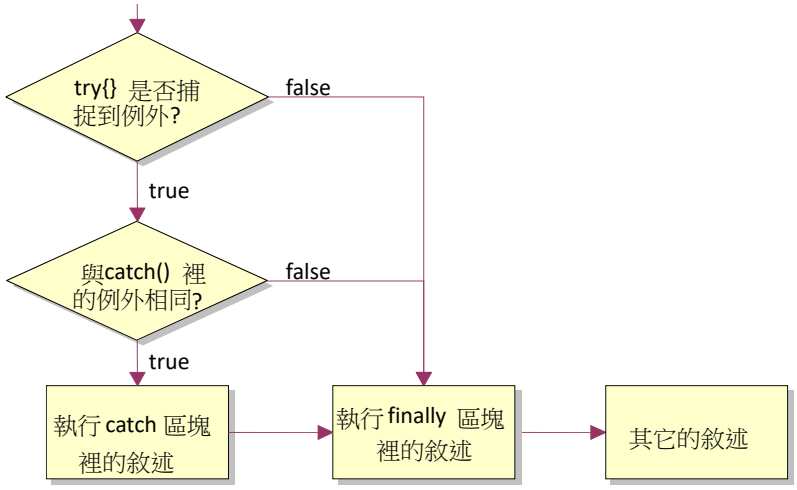
catch區塊

finally區塊

格式13.1.1
例外處理的語法

13-7

下圖繪出了例外捕捉的流程圖：



13-8

下面是例外處理的範例：

```
/* app13_2 OUTPUT----------
index out of bound!!
this line is always executed!!
end of main() method!!
-------------------------*/
```

```
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!!"); //繼續執行
20    }
21  }
```

程式監控區
(try把會出錯的
程式碼包起來)

錯誤的時候
處理的區塊

必定執行

13-9

如果捕捉到例外，**Java會利用例外類別建立一個類別變數e：**

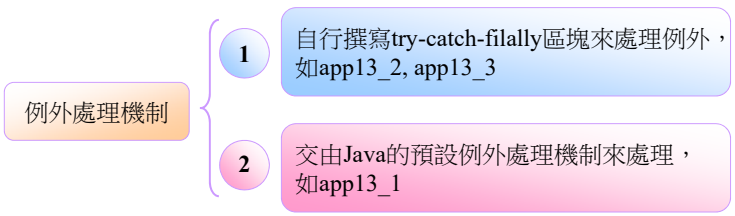
```
01  // app13_3, 例外訊息的擷取
02  public class app13_3
03  {
04    public static void main(String args[])
05    {
06      try
07      {
08        int arr[]=new int[5];
09        arr[10]=7;
10      }
11      catch(ArrayIndexOutOfBoundsException e)
12      {
13        System.out.println("index out of bound!!");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

```
/* app13_3 OUTPUT----------
index out of bound!!
Exception=java.lang.ArrayIndexOutOfBoundsException
end of main() method !!
----------*/
```

13-10

13.1.4 例外處理機制的回顧

下圖繪出了例外處理機制的選擇流程：



請先閱讀下面的程式碼，嘗試了解其中每一行的意義，並試著回答接續的問題：

```
02 public class ex13_1
03 {
04   public static void main(String args[])
05   {
06    int num=12,den=0;
07    int ans=num/den;
08    System.out.println("end of main() method !!");
09   }
10 }
```

(a) 在**編譯**此程式時，會不會有錯誤訊息產生？
(b) 執行**ex13_1** 後，系統會拋出如下的例外：**Exception in thread "main" java.lang.ArithmeticException: / by zero**試說明這個例外的涵義，並指出為什麼會產生這個例外。
(c) 於本例中，程式碼的第**8** 行是否會被執行？試說明會或不會的原因。

13.1 課堂練習
例外的基本觀念

ex13_1_1.java

試修改習題 ex13_1.java的程式碼，加入 if-else 的判斷式，若 den 為 0，則跳過除法的運算，並印 出 "除數為 0" 的字串，若 den 不為 0，則進行除法運算。

```
/* output--------------
  除數為 0
  end of main() method!!
--------------------*/
```

14

13.1 課堂練習
例外的基本觀念

ex13_1_2.java

延續上一題，加入try-catch 的處理，使結果如下。

```
/* output--------------
數值錯誤
Exception:java.lang.ArithmeticException: / by zero
end of main() method!!
---------------------*/
```

15

13.1 回家作業
例外的基本觀念

hw13_1_1.java

試修改下面的程式碼，加入 if-else 的判斷式，若除數為 0，則跳過除法的運算，並印出“除數為 0” 的字串，若除數不為 0，則進行除法運算：

```
public class hw13_1_1
{
 public static void main(String args[])
 {
  int num=5,d1=3,d2=0,d3=0,d4=1;
  System.out.println("num/d1= "+num/d1);
  System.out.println("num/d2= "+num/d2);
  System.out.println("num/d3= "+num/d3);
  System.out.println("num/d4= "+num/d4);
  System.out.println("end of main() method!!");
 }
}
```

```
/* output--------------
num/d1= 1
除數為 0
除數為 0
num/d4= 5
end of main() method!!
---------------------*/
```

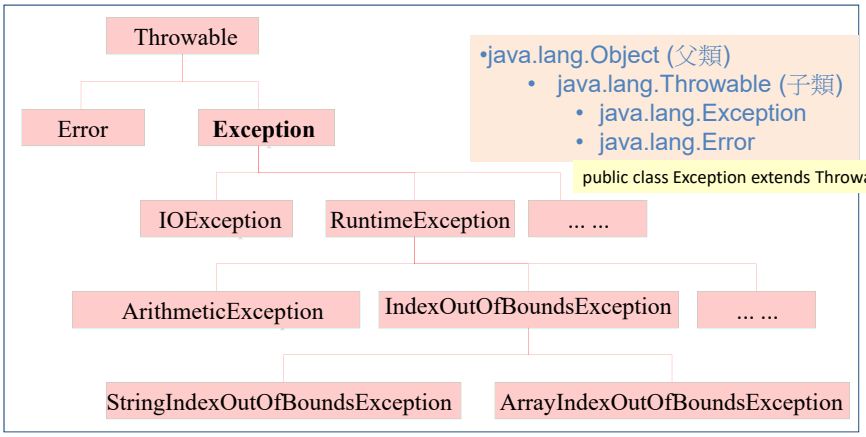
16

13-11

13.2 例外類別的繼承架構

例外類別可分為兩大類：java.lang.Exception 與java.lang.Error。

下圖為Throwable類別的繼承關係圖：



https://docs.oracle.com/en/java/javase/17/docs/api/index.html

Exception and Error說明

- Exception:
  - 通常指程式運行時所出現的可預料之意外狀況, 基本上都要進行 catch , 進行相對應處理, 如 IOException.
- Error:
  - 指在正常情況下, 不太可能出現的問題, 絕大部分的 Error 都會導致程式 (e.g. JVM 本身) 處於一種不正常且不可恢復的狀態. 所以對於這種情況, 不太需要去 catch , 因為也沒什麼意義.
  - 常見的如 OutOfMemoryError / StackOverflowError 這些, 都是繼承自 Error.

13-12

當例外發生時，catch() 程式區塊會接收由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
}
```

13-13

使用try捕捉一種以上的例外

若要捕捉ArithmeticException 和 ArrayIndexOutOfBoundsException 兩個例外，可以用如下的語法來撰寫：

```
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  }
```

捕捉所有的例外

13-14

不論例外類別為何，所有例外都想捕捉它的話，則可在catch() 的括號內填上Exception例外：

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

下面的程式碼裡存在有兩個錯誤，第一個是除數為**0**，第二個是陣列索引值超出了範圍。試在此程式碼內加入**if-else** 敘述與**break**，使得當這兩種錯誤有任何一種發生時，程式便會停止執行，並印出 **"程式執行有誤"** 字串。

ex13_2_1

```
01 // ex13_2_1
02 public class ex13_2_1
03 {
04  public static void main(String args[])
05  {
06    int num=12;
07    int den[]={12,0,3,0,0,4};
08
09   for(int i=0;i<10;i++)
10    System.out.println("ans="+num/den[i]);
11   }
12 }
```

```
/* output----------
ans=1
程式執行有誤
----------*/
```

ex13_2_2

- 試修改ex13_2_1 的程式碼，使用**try-catch** 區塊來捕捉由錯誤而產生的例外。

```
/* output----------
ans=1
除數為 0
----------*/
```

```
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 }
```

13.2 回家作業
例外類別的繼承架構

hw13_2_1.java

試修改習題 ex13_2_1 的程式碼，使用 try-catch 區塊來捕捉由錯誤而產生的例外，其中 catch() 可捕捉所有可能由系統拋出的例外，並印出 "捕捉到例外了" 字串（提示：由 catch() 來捕獲 Exception 例外類別的物件）。

```
/* output----------
ans=1
捕捉到例外了
----------*/
```

24

13.2 回家作業
例外類別的繼承架構

hw13_2_2.java

請先閱讀下面的程式碼，嘗試了解其中每一行的意義：

```
public class ex13_9
{
  public static void main(String args[])
  {
    int num=3;
    double den=0.0;
    System.out.println(num+"/"+den+"="+num/den);
  }
}
```

(a) 在編譯此程式時，會不會有錯誤訊息產生？

(b) 您得到什麼樣的結果？為什麼？

25

Exception handling 補充說明

```
package temp;
import java.io.*;
public class test1
{
 public static void main(String args[])
 {
try
{
    System.out.println("Opening FileInputStream");
    FileInputStream f = new FileInputStream("abc.txt"); // assume this operation generate FileNotFoundException
    System.out.println("File Opened");
}

catch (FileNotFoundException e1)
{
    System.out.println("FileNotFoundException caught");
}

catch (Exception e2)
{
    System.out.println("Exception caught");
}

finally
{
    System.out.println("Execute finally block");
}
 }
}
```

```
Opening FileInputStream
FileNotFoundException caught
Execute finally block
```

Exception handling 補充說明

```
package temp;
import java.io.*;
public class test1
{
 public static void main(String args[])
 {
try
{
    System.out.println("Opening FileInputStream");
    FileInputStream f = new FileInputStream("abc.txt"); // assume this operation generate FileNotFoundException
    System.out.println("File Opened");
}

catch (Exception e2)
{
    System.out.println("Exception caught");
}

catch (FileNotFoundException e1)
{
    System.out.println("FileNotFoundException caught");
}

finally
{
    System.out.println("Execute finally block");
}
 }
}
```

```
Exception in thread "main" java.lang.Error:
Unresolved compilation problem:
Unreachable catch block for
FileNotFoundException. It is already handled
by the catch block for Exception

at temp.test1.main(test1.java:18)
```

Exception handling 補充說明

- try-catch-**finally** ：finally區塊裡面的程式碼大多用來作**資源回收,關閉資料庫連線, 或清理資料結構的工作,以確保不論有無發生狀況,程式都能繼續正常執行。**

```
import java.io.*;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.Scanner;
public  class test1{
public static void main(String args[]) throws IOException
{
String s=readFile("C:\\Java\\list.txt");
System.out.println(s);
} public static String readFile(String name) throws FileNotFoundException {
        StringBuilder builder = new StringBuilder();
        Scanner scanner = null;
        try {
            scanner = new Scanner(new FileInputStream(name));
            while (scanner.hasNext()) {
                builder.append(scanner.nextLine());
                builder.append('\n');
            }
        } finally {
            if(scanner != null) {
                scanner.close();
            }
        }
        return builder.toString();
    }}
```

13.3 拋出例外(throw 與throws的區別)

13-15

- throw：就是程式中自己處理異常，處理時可由自己拋出異常(try catch)或系統判定拋出。
- throws：一個方法裡面不想有任何的異常處理，則必須對該方法進行宣告有可能產生的所有異常。

(1)於程式中拋出例外

throw拋出一個物件

- 由程式撰寫，來決定如何拋出(有寫 throw, app13_4)
- 由系統自己判定拋出(try catch,沒寫throw, app13_5)

(2) 指定method拋出例外

throws描述某個method可以拋出例外

- 寫在同一個class中(app13_6)
- 寫在不同一個class中 (app13_7)

13.3.1於程式中拋出例外

13-16

於程式中拋出例外時，要用到throw這個關鍵字，其語法如下：

```
throw 由例外類別所產生的物件;
```

格式13.3.1
例外處理的語法

app13_4是於程式中拋出例外的範例：

13-17

```
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
----------------------------------*/
```

app13_5是讓系統自動拋出例外的驗證：

13-18

```
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    }
17  }
```

```
/* app13_5 OUTPUT----------------------------------
java.lang.ArithmeticException throwed: / by zero throwed
-------------------------------------------------------*/
```

13.3.2指定method拋出例外

13-19

若是要由method拋出例外，method必須以下面的語法來定義：

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

格式13.3.2
由method拋出例外

app13_6是指定由method來拋出例外的範例。

13-20

```
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
14     {
15       aaa(4,0);
16     }
17     catch(ArithmeticException e)
18     {
19       System.out.println(e+" throwed");
20     }
21   }
22 }
```

```
/* app13_6 OUTPUT----------
java.lang.ArithmeticException: / by zero throwed
----------*/
```

app13_7說明了如何從不同類別裡的method裡拋出例外：

13-21

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

試修改ex13_3_1，將**6~7** 行的程式碼寫在一個可拋出**ArithmeticException** 的 **test()method** 內，並在**main() method** 內撰寫程式碼來捕捉由**test()** 所拋出的例外（請參考**app13_6** 的寫法）。

ex13_3_1

```
01 // 例外訊息的擷取
02 public class ex13_3_1
03 {
04  public static void main(String args[])
05  {
06   int num=12,den=0;
07   int ans=num/den;
08   System.out.println("end of main() method !!");
09  }
10 }
```

```
/* output----------
除數為 0
end of main() method!!
----------*/
```

試將ex13_3_2所撰寫的**test() method** 改寫在一個獨立的**Ctest** 類別內，使得**ArithemeticException** 例外是由**Ctest** 類別內的**test() method** 所拋出（請參考**app13_7**的寫法）。

ex13_3_2

```
01 // 例外訊息的擷取
02 public class ex13_3_2
03 {
04 public static void main(String args[])
05 {
06 int num=12,den=0;
07 int ans=num/den;
08 System.out.println("end of main() method !!");
09 }
10 }
```

```
/* output----------
除數為 0
end of main() method!!
----------*/
```

13.3 回家作業
拋出例外

hw13_3_1.java

- 建立一長度為3的陣列，試列印陣列第4個值，利用try-catch 並捕捉例外訊息。
- 最後程式結束，請列印 “Out of the block” 。

```
/* output----------
Exception thrown  :java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 2
Out of the block
----------*/
```

38

13.3 回家作業
拋出例外

hw13_3_2.java

建立一個包含一個power(int,int)方法的myCalculator類別。這個power方法可以傳入n、p兩個整數作為參數，並計算n^p。若n或是p為負數，這個方法將會輸出一個例外，訊息為「n and p should be non-negative」。

*使用Scanner輸入

*在power方法中判斷n,p是否為負數，如為負數:

throw new Exception("n and p should be non-negative");

```
/* output----------
輸入兩數值:
12 2
144
----------*/
```

```
/* output----------
輸入兩數值:
-2 5
java.lang.Exception: n and p should be non-negative
----------*/
```

39

13.4 自己撰寫例外類別

13-22

自行撰寫例外類別的語法如下：

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

格式13.4.1
撰寫自訂例外類別的語法

下面的範例說明了如何定義自己的例外類別，以及使用它們：

13-23

```
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 {
08   private double radius;
09   public void setRadius(double r) throws CCircleException //method拋出例外
10   {
11     if(r<0)
12     {
13       throw new CCircleException();  // 拋出例外
14     }
15     else
16       radius=r;
17   }
18   public void show()
19   {
20     System.out.println("area="+3.14*radius*radius);
21   }
22 }
```

CCircleException沒有任何敘述，但因繼承關係，所以此類別已經具備了基本例外處理的功能。

建構物件時，成員會進行初始化，如果沒有指定初始值，則會使用預設值初始化。例如：整數的初始值=0，double的初始值=0.0, float=0.0f, reference(String)=null

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

```
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 }
```

13.5 拋出輸入/輸出的例外類別

13.5.1拋出輸入型態不合的例外

- 程式中預設要讓使用者輸入整數，使用者卻輸入英文字母，會拋出InputMismatchException例外後中斷執行

```
01 // app13_9, 捕捉 InputMismatchException 例外
02 import java.util.*;
03 public class app13_9
04 {
05   public static void main(String args[])
06   {
07     int num;
08     Scanner scn=new Scanner(System.in);
09     try
10     {
11       System.out.print("請輸入一個整數: "); // 輸入整數
12       num=scn.nextInt();
13     }
14     catch(Exception e)                    // 補捉所有的例外
15     {
16       System.out.println("拋出"+e+"例外"); // 印出例外的種類
17     }
18   }
19 }
```

執行過程中刻意輸入英文字母，方便觀察拋出的例外種類

```
/* app13_9 OUTPUT----------
請輸入一個整數: k
拋出 java.util.InputMismatchException 例外
-----------------------------------------*/
```

InputMismatchException的英文分解，即Input Mismatch Exception，表示輸入的資料型態不合之義

42

13.5.2 拋出IOException例外類別

13-25

會拋出IOException例外的method，其例外處理的方式有兩種。

第一種是直接由main()拋出例外，app3_13(如下)所採用的就是這種機制：

```
01  // app3_13,由鍵盤輸入字串
02  import java.io.*;
03  public class test
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

第一種直接由main()拋出例外的範例

```
package ex;
  import java.io.*;  // 載入java.io類別庫裡的所有類別
  public class ex
  {
    public static void main(String args[]) throws IOException
    {
      BufferedReader buf=new BufferedReader(new InputStreamReader(System.in));
      String str;
      System.out.print("Input a string: ");
      str=buf.readLine(); // 將輸入的文字指定給字串變數str存放

      System.out.println("string="+str);  // 印出字串
    }
  }
```

第二種方式是撰寫try-catch區塊來捕捉拋出的例外

13-26

如下面的範例：

```
01 // app13_10,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!
-------------------------*/
```

ex13_4_1

- 試修改**app13_8**，使得當半徑大於**100** 時，會拋出**RadiusTooLarge** 例外，且半徑小於**0** 時，拋出**RadiusIsNegative** 例外。

```
/* output-----------------
RadiusTooLarge throwed
area=0.0
------------------------*/
```

13.4 回家作業 拋出例外

hw13_4_1.java

試撰寫一程式，可以由鍵盤輸入一字串。若字串之值為 "520"，則拋出 Exception520 這個例外物件，並印出下面的字串：

"這是由字串 520 所引起的例外" 如果輸入的字串不為 "520"，則印出原來輸入的字串。

```
/* output----------
Input a string:520
這是由字串520所引起的例外
----------*/
```

47

13.4 回家作業 拋出例外

hw13_4_2.java

試撰寫一程式，可以由鍵盤輸入一整數字串。請先把整數字串轉換成整數，再依其值 來撰寫例外：

(a) 若整數之值小於 10，則拋出 IntegerTooSmall 這個例外物件，並印出下面的字串：

"您輸入的整數的值太小"

(b) 若整數之值大於 70，則拋出 IntegerTooLarge 這個例外物件，並印出下面的字串：

"您輸入的整數的值太大"

(c) 若整數之值介於 10 和 70 之間，則印出原來的數值。

```
/* output----------
Input an integer:87
您輸入的整數值太大
----------*/
```

49

13.4 回家作業 自己撰寫例外類別

hw13_4_3.java

下面的程式，是印出 1~n 之間的奇數值。請先閱讀下面的程式碼，嘗試了解其中每一行的意義：

```
public class hw13_4_1
{
  public static void main(String args[]) {
    int n=11;
    for(int i=1;i<=n;i+=2)
      System.out.print(i+" ");
    System.out.println();
  } }
```

```
/* output--------------
1 3 5 7 9 11
--------------------*/
```

(a) 請將印出奇數值部分的程式碼，改成 void odd(int n) method。

(b) 當 n 小於等於 0 時，請由 method 拋出 IllegalArgumentException 例外，並印出 "n 值小於等於 0,無法處理" 字串。

(c) 當 n 為偶數值，請由 method 拋出 NotOddException 例外，並印出"n 值為偶數, 無法處理" 字串。

50

13.4 回家作業 自己撰寫例外類別

hw13_4_4.java

三角形的三個邊為 a、b、c。當 a=b=c 時，即構成正三角形。下面的程式，可以判斷是否為正三角形：

```
public class hw13_4_2  {
  public static void main(String args[]) {
    int a=3;
    int b=3;
    int c=3;
    if((a+b)<c || (a+c)<b || (b+c)<a)
      System.out.println("不構成三角形");
    else if(a==b && a==c && b==c)
      System.out.println("這是正三角形");
    else
      System.out.println("這不是正三角形");
  }}
```

```
/* output--------------
a=5, b=5, c=5
這是正三角形
--------------------*/
```

請將判別三角形的程式碼撰寫到void triangle(int a,int b,int c) method，並利用try-catch 區塊捕捉自訂的例外，這些自訂的例外皆由 triangle() method 拋出。請完成下面各題的要求：

(a) 若此三邊不能構成一個三角形，則拋出自訂的 NotTriangle，並印出 "不構成三角形" 字串。

(b) 若此三角形為正三角形，則拋出自訂的 EquilateralTriangle，並印出 "這是正三角形" 字串。

(c) 若此三角形不為正三角形，則拋出自訂的 NotEquilateralTriangle，並印出 " 這不是正三角形" 字串。

51