



JAVA例外處理 (Exception Handling)

授課老師：陳俊豪

E-mail: chchen6814@gmail.com

Outline

- Bug 的分類
- Error、Exception 與 Throwable類別
- try、catch 區塊結構
- finally 區段
- Call Stack 機制
- 利用 throw 丟出例外
- 利用 throws 自定例外方法
- 利用繼承Exception類別自定例外類別
- 斷言錯誤(AssertionError)

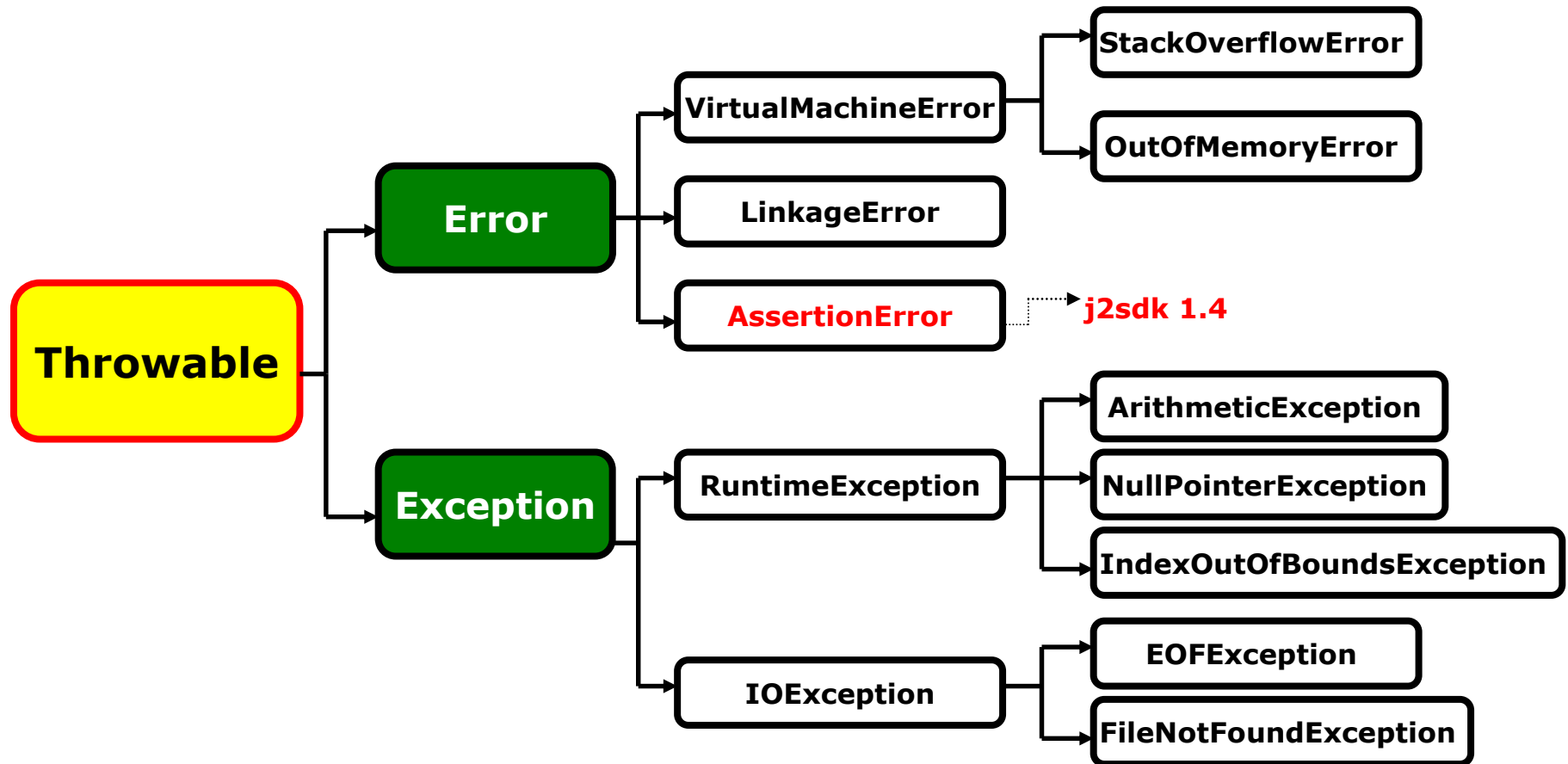
Bug 的分類

- “Bug”中文的解釋即是臭蟲、小蟲的意思
- 程式上的錯誤依性質可分為以下 3 種：
 - ▣ 程式語法上的錯誤：
 - `char c = “SCJP”;`
 - `File f = new File(“xxx”); // 忘了 import java.io.*;`
 - ▣ 執行時期的錯誤：
 - 陣列元素索引值超出最大範圍。
 - 整數除以 0。
 - ▣ 邏輯的錯誤：
 - 利息的計算公式、公司獎金配發比例以及是否要進位(四捨五入) ...等。

Errors、Exception 與 Throwable 類別

- Java 中用來處理錯誤的類別分為：
 - ▣ Throwable為Error與Exception的父類別
 - Error
 - 通常泛指的是系統本身發出的錯誤訊息，也有可能是程式所造成的
 - e.g., 記憶體不足
 - Exception
 - 是一個不正常的程式(當下而言)在執行期間所觸發的事件
 - e.g., 數學上的除零錯誤

常見錯誤類別的樹狀圖



例外處理法try、catch 區塊結構

□ try

- ▣ 撰寫程式的時候，可將會發生例外狀況(Exception)的程式碼用 try 區塊包住

□ catch

- ▣ 表示try區塊有發生Exception時，將利用 catch 區塊攔截錯誤訊息(例外)並執行此區塊內的程式碼(處理)

□ try – catch → Error handling

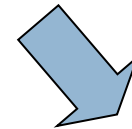
try – catch 語法

```
try {  
    ...// Statement  
}  
catch(FileNotFoundException e1) {  
    ...// Statement  
}  
catch(Exception e2) {  
    ...// Statement  
}
```

Example: 沒有try catch時

```
import static java.lang.System.*;
public class ErrorHandling {
    static int numerator = 20; // 分子
    static int[] denominator = {0, 2, 4}; // 分母
    static String answer; // 計算結果
    public static void main(String[] args) {
        calc(0);
        out.println(numerator + " / " + denominator[0] + " = " + answer);
        calc(2);
        out.println(numerator + " / " + denominator[2] + " = " + answer);

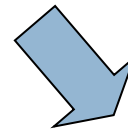
        out.println("計算完畢 !");
    }
    public static void calc(int index) {
        double ans = 0;
        ans = numerator / denominator[index];
        answer = String.valueOf(ans);
    }
}
```



會發生整數不得除以0的錯誤，
之後終止程式。

Example: 加了try catch之後

```
import static java.lang.System.*;
public class ErrorHandler2 {
    .....
}
public static void calc(int index) {
    double ans = 0;
    try {
        ans = numerator / denominator[index];
        answer = String.valueOf(ans);
    }
    catch(ArithmeticException e) {
        System.out.println("產生了數學錯誤, 原因是 : " + e.getMessage());
        System.out.println("詳細錯誤 : ");
        e.printStackTrace(out);
        answer = "無法計算";
    }
}
```



有try catch處理"整數不得除以0的錯誤",
因此, 程式可以繼續執行。

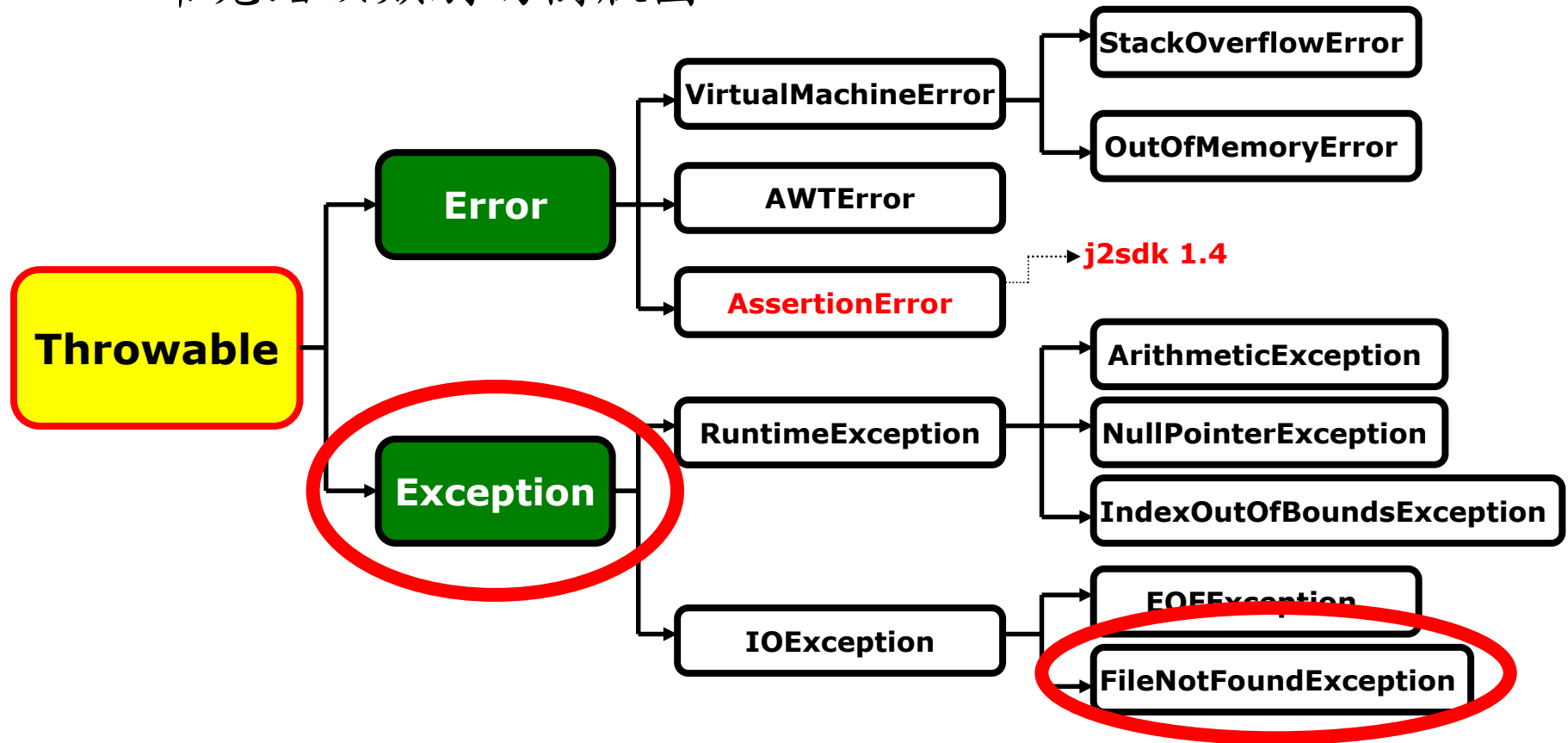
Example: 如果將程式改成cal(99)呢?

- 會產生java.lang.ArrayIndexOutOfBoundsException的例外錯誤
- 必須再寫一個catch來抓住它

```
try {  
    ans = numerator / denominator[index];  
    answer = String.valueOf(ans);  
}  
catch(ArithmeticException e) {  
    System.out.println("產生了數學錯誤, 原因是 : " + e.getMessage());  
    System.out.println("詳細錯誤 : ");  
    e.printStackTrace(out);  
    answer = "無法計算";  
}  
catch(ArrayIndexOutOfBoundsException e) {  
    System.out.println("產生了錯誤, 原因是 : " + e.getMessage());  
    System.out.println("詳細錯誤 : ");  
    e.printStackTrace(out);  
    answer = "無法計算";  
}
```

直系的父類別 VS. 直系的子類別

□ 常見錯誤類別的樹狀圖



try – catch: 多個例外捕捉撰寫順序

□ try – catch 語法：

```
try {  
    ...// Statement  
}  
catch(FileNotFoundException e1) {  
    ...// Statement  
}  
catch(Exception e2) {  
    ...// Statement  
}
```

檔案未發現例外(直系的子類別)
*順序上，子類別需放在上面

所有例外(直系的父類別)
*順序上，父類別需放在下面

finally 區段

- finally 是 Java 的關鍵字
 - ▣ 不論程式是否有例外(Exception)發生，在程式跳離 try-catch 區塊之前一定會執行的程式區段
 - ▣ 換句話說，finally 區段是 **always execute** 的程式區段
- 有沒有甚麼方法讓 Java 程式不執行 finally 區段？
 - ▣ **System.exit()**
 - 在 finally 區段前執行了 System.exit() 函式，強迫執行中的程式結束
 - ▣ **關機**
 - 程式在執行 finally 區段前關機。

finally 語法

```
try {  
    ...// Statement  
}  
catch(ExceptionType1 e1) {  
    ...// Statement  
}  
finally {  
    ...//在跳離try...catch前最後所必須要執行的程式碼  
}
```

Example: finally

```
import static java.lang.System.*;
public class ErrorHandling2 {
    //需撰寫程式碼請參閱前面
}
public static void calc(int index) {
    double ans = 0;
    try {
        ans = numerator / denominator[index];
        answer = String.valueOf(ans);
    }
    catch(ArithmeticException e) {
        System.out.println("產生了數學錯誤, 原因是 : " + e.getMessage());
        System.out.println("詳細錯誤 : ");
        e.printStackTrace(out);
        answer = "無法計算";
    }
    finally{
        System.out.println("程式執行結束");
    }
}
}
```

try catch finally 正確用法

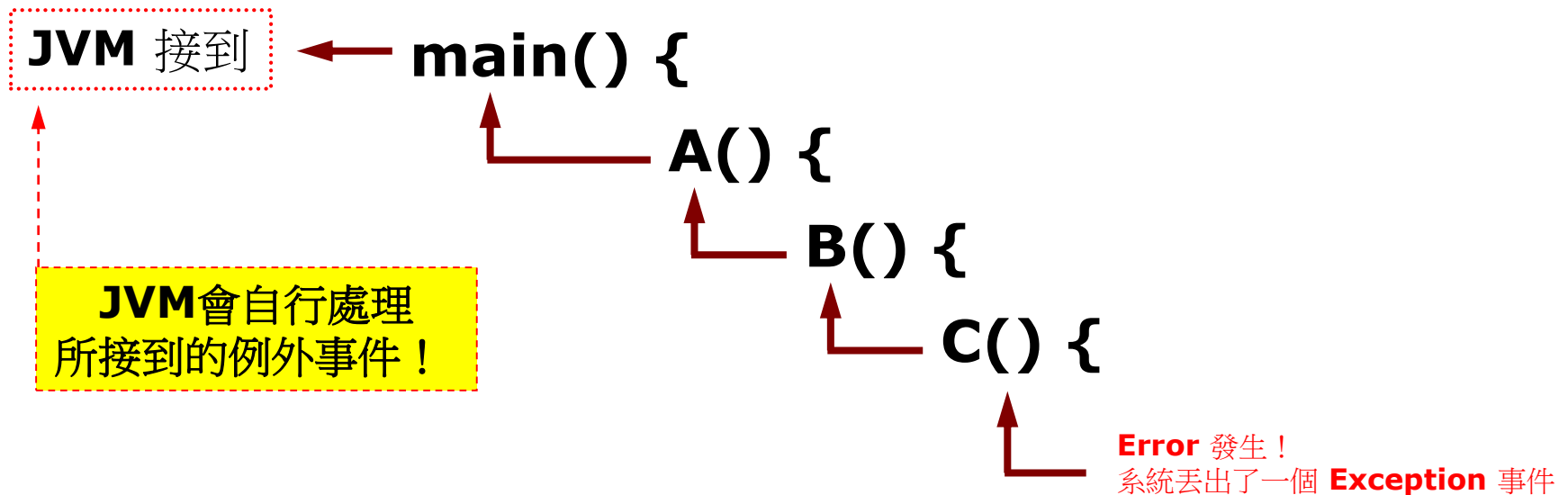
□ 在 Java 語言中

- ▣ try 程式區塊可與 catch 或 finally 區段搭配使用
- ▣ 不過 catch 與 finally 必須要搭配 try 區段才可使用

try	catch	finally	編譯
○	○	○	成功
○	○	×	成功
○	×	×	失敗
○	×	○	成功
×	○ 或 ×	○ 或 ×	(失敗)

Call Stack 機制

- 將例外事件一層層的往上丟直到被處理為止，
這整個過程就稱為 Call Stack Mechanism



Call Stack 機制(Cont.)

□ Call Stack 與 try-catch

```
main() {
```

```
    A() {
```

```
        B() {
```

```
            try {
                ↑ C() {
                    ↑ ...// 丟出一個 Exception
                }
            }
            catch (Exception) {
                ...// 錯誤處理程式！
            }
```

throw與throws關鍵字

- 探討如何撰寫自己的例外錯誤類別
- 甚至是商業邏輯(business logical)的錯誤
 - ▣ 例如
 - 設計LoggingException(登入例外錯誤)
 - 設計IDNumberException (身分證號例外錯誤)
 - 設計BusinessTransactionException (交易例外錯誤)
 - etc.

throw 的程式語法

□ throw 關鍵字

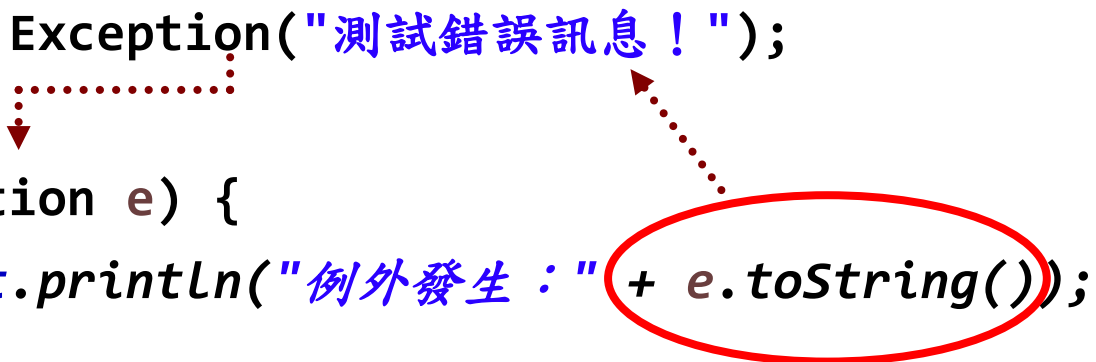
- ▣ 用來呼叫或傳遞一個例外
- ▣ 可以利用它在程式中觸發一個例外錯誤

□ throw 的程式語法

- ▣ throw 例外物件變數;
 - ▣ 丟出一個例外物件變數
- ▣ throw new Exception(錯誤訊息字串);
 - ▣ 丟出一個匿名的例外物件

throw 範例程式

```
public class throwSample {  
    public static void main(String[] args) {  
        try {  
            throw new Exception("測試錯誤訊息!");  
        }  
        catch(Exception e) {  
            System.out.println("例外發生：" + e.toString());  
        }  
    }  
}
```



throws 自定例外方法

- 若Java方法會產生例外錯誤
 - ▣ 可在宣告方法時加上 throws 關鍵字來修飾該方法
- 語法
 - ▣ 方法名稱(參數列) throws ext1, ext2 ... {}
 - ▣ throws 一定要擺在方法參數列後面以及“{“前
- 範例

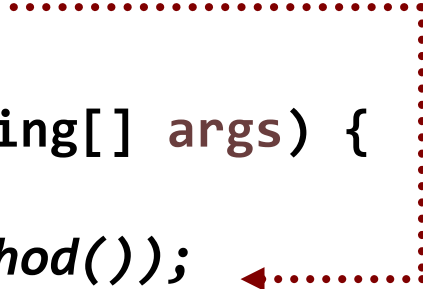
```
public static String aMethod() throws IOException {  
    return "測試錯誤訊息！";  
}
```

表示此方法將會產生輸出輸入例外

throws 範例程式

□ throws 範例

```
public class throwsSample {  
    public static String aMethod() throws Exception {  
        return "測試錯誤訊息！";  
    }  
    public static void main(String[] args) {  
        try {  
            System.out.println(aMethod());  
        }  
        catch(Exception e) {}  
    }  
}
```



Example

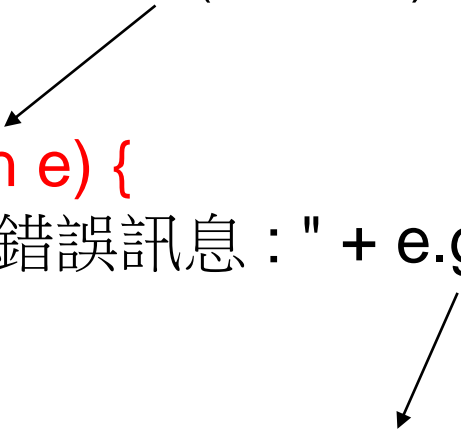
```
import static java.lang.System.*;
public class ErrorHandling3 {
    static int numerator = 20; // 分子
    static int[] denominator = {0, 2, 4}; // 分母
    static String answer; // 計算結果
    public static void main(String[] args) {
        try { calc(0); }
        catch(Exception e) {out.println("錯誤訊息 : " + e.getMessage()); }
        out.println(numerator + " / " + denominator[0] + " = " + answer);
        out.println("計算完畢 !");
    }
    public static void calc(int index) throws Exception{
        double ans = 0;
        if ((index == 0) || (index >= denominator.length)) {
            answer = "無法計算";
            throw new Exception("denominator[] 的索引值" + "不得為 " + index);
        }
        ans = numerator / denominator[index];
        answer = String.valueOf(ans);
    }
}
```



```
try {  
    calc(0);  
} catch(Exception e) {  
    out.println("錯誤訊息：" + e.getMessage());  
}
```

calc(int index) throws Exception

throw new Exception("denominator[] 的索引值" + "不得為 " + index);



註: throw 所丟出的例外類別必須是 throws 的子類別

合法:

```
public static void calc(int index) throws Exception{  
    throw new IOException();  
}
```

合法:

```
public static void calc(int index) throws IOException, RuntimeException{  
    throw new IOException();  
}
```

不合法:

```
public static void calc(int index) throws IOException {  
    throw new Exception();  
}
```

Exception 建構子說明

- Exception 建構子
 - ▣ 我們可以利用 Exception 建構子來自訂錯誤訊息文字。
- Exception 類別中建構子有哪些定義？
 - ▣ Exception()
 - ▣ 預設建構子(Default Constructor)。
 - ▣ Exception(String message)
 - ▣ 自行定義的錯誤訊息傳入。
 - ▣ Exception(String message, Throwable cause)
 - ▣ 將自行定義的錯誤訊息與導致發生的原因傳入。
 - ▣ Exception(Throwable cause)
 - ▣ 將導致發生的原因傳入。

利用繼承Exception 類別自定例外類別

- 若要在程式中自定例外類別，可透過繼承(extends)完成
- 宣告語法：
 - ▣ 類別名稱 extends 基礎例外類別名稱 {}
- 撰寫自定例外的 4 個步驟：
 - ▣ 1. 產生自定例外類別
 - ▣ 2. 在宣告方法時加入 throws 宣告
 - ▣ 3. 在方法中使用 throw new method_name() 以觸發例外事件
 - ▣ 4. 將此方法放入try...catch程式區塊內

Example: 步驟1

□ 1. 產生自定例外類別

```
class MemberIDException extends Exception{  
    public MemberIDException(String mID) {  
        super("會員證號 " + mID + " 驗證失敗!");  
    }  
    public void contactWith() {  
        System.out.println("請連絡服務人員 : Tel (xx)-(xxxx-xxxx) !");  
    }  
}
```

Example: 步驟 2, 3, 4

```
public class ErrorHandling4 {  
    public static void main(String[] args) {  
        boolean verify = true;  
        try {  
            checkMemberID("1234567890");  
        }  
        catch(MemberIDException e) {  
            out.println("錯誤訊息：" + e.getMessage());  
            e.contactWith(); // 自訂例外類別之自訂方法  
            verify = false;  
        }  
        if(verify) out.println("會員證號驗證成功 !");  
        else out.println("會員證號驗證失敗 !");  
    }  
    public static void checkMemberID(String mID) throws MemberIDException{  
        if (mID.length() != 5) {  
            throw new MemberIDException(mID);  
        }  
    }  
}
```

4. 將此方法放入**try...catch**程式區塊內。

2. 在宣告方法時加入 **throws** 宣告。

3. 在方法中使用 **throw new method_name()** 以觸發例外事件。

斷言錯誤(AssertionError)

□ 斷言(assert)

- ▣ 用來彌補 try-catch 的不足，讓程式更堅固
- ▣ 用在“你認為”你的程式不可能會產生錯誤的地方
- ▣ 如assert發生，程式會丟出 AssertionError

□ 例如

- ▣ 一些在程式邏輯的觀點來說是正常但在商業邏輯上卻會發生很大的問題
 - 例如：商品購買數量出現負數、成績為負數等

assert 語法

```
assert (Boolean 運算式或 Boolean 涵式);  
assert (Boolean) : “陳述子句” ;
```



當程式retrun false 時執行

□ 編譯 assert 程式

✓ javac -source 1.4 [JavaFile]

□ 執行 assert 程式

✓ java -enableassertions [ClassName]



enable assertion

Run -> Run Configurations -> Arguments -> VM arguments 中加上 -enableassertions

assert 範例

```
01. public class AssertionSample {  
02.     public static void main(String[] args) {  
03.         int score = 70;  
04.  
05.         if (score >=60) {  
06.             System.out.println("及格!");  
07.         }  
08.         else if (score < 60) {  
09.             System.out.println("不及格!");  
10.         }  
11.     }  
12. }
```

assert 範例 (Cont.)


□ assert 範例

```
01. public class AssertionSample {  
02.     public static void main(String[] args) {  
03.         int score = 30;  
04.  
05.         if (score >=60) {  
06.             System.out.println("及格!");  
07.         }  
08.         else if (score < 60) {  
09.             System.out.println("不及格!");  
10.         }  
11.     }  
12. }
```

assert 範例 (Cont.)

□ assert 範例

```
1. public class AssertionSample {  
2.     public static void main(String[] args) {  
3.         int score = -10;  
4.  
5.         if (score >= 60) {  
6.             System.out.println("及格!");  
7.         }  
8.         else if (score < 60) {  
9.             System.out.println("不及格!");  
10.        }  
11.    }  
12. }
```



assert 範例 (Cont.)

```
public class AssertionSample {  
    public static void main(String[] args) {  
        int score = -10;  
        assert (score >= 0) : "成績錯誤~";  
        if (score >= 60) {  
            System.out.println("及格!");  
        } else if (score < 60) {  
            System.out.println("不及格!");  
        }  
    }  
}
```

當結果為false時會產生AssertionError




assert 範例 2

```
public class AssertionSample2 {  
    public static void main(String[] args) {  
        boolean MUST_BE_TRUE = true;  
        while(MUST_BE_TRUE){  
            //block of code  
        }  
        assert(false):"不可能執行到這裡!";  
    }  
}
```

換句話說, 如果發生 **AssertionError** 則表示, **while-loop** 有問題

assert 範例 3

```
public class AssertionSample {  
    public static void main(String[] args) {  
        char sex = '1';  
        switch(sex){//假設已取得身分證第二碼  
            case '1': System.out.println("男"); break;  
            case '2': System.out.println("女"); break;  
            default : assert false: "性別有誤!";  
        }  
    }  
}
```



換句話說, 如果發生**AssertionError**則表示, **switch**有問題