第11章
例外處理、斷言與資源回收

資訊科技系
林偉川

11-1-1 程式錯誤的分類

程式的錯誤可以依照性質分成三種：

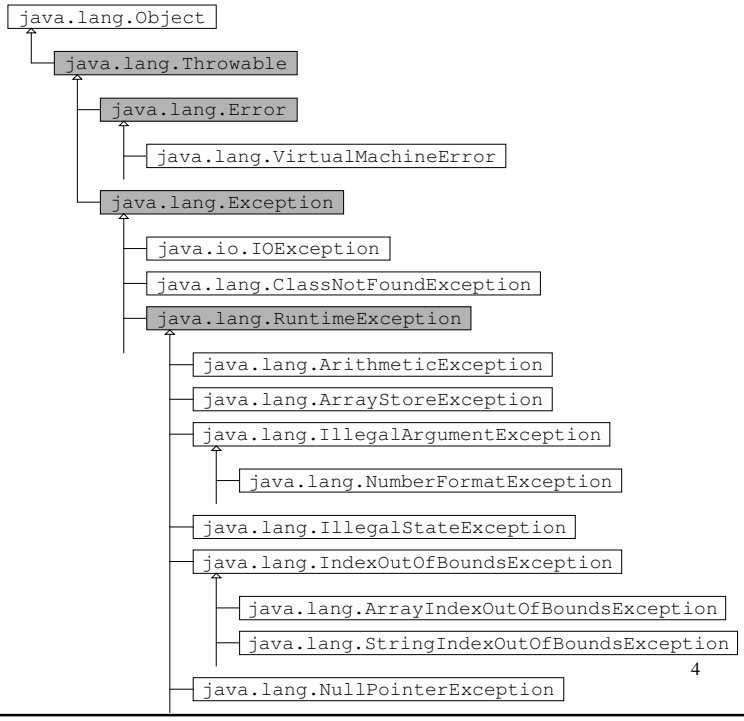
- **編譯期錯誤**：編譯時所發生的錯誤，經常是**語法錯誤**。
- **執行期錯誤**：在編譯程式的過程中不容易發現，常在程式執行的過程中發生的錯誤。
- **邏輯錯誤**：程式設計者在設計程式時，所發生的邏輯上的問題。

2

11-1-2 Java例外的繼承關係

- 錯誤或例外都是java.lang.**Throwable**的延伸類別
  - **Error**及其延伸類別：其為嚴重的錯誤，通常捕捉到也無法處理，因此也很少去捕捉。
  - **Exception**及其延伸類別(不包含RuntimeException及其延伸類別)：這類例外在一般情況很可能發生，大多屬於環境問題。
  - **RuntimeException**及其延伸類別：此類例外為程式的執行期錯誤，例如，某數除以0、陣列索引值超出範圍等。

3



4

11-1-3 常見的RuntimeException

- 常見的RuntimeException(Unchecked Exception)

執行期例外	說明
ArithmeticException	數學運算時的例外。例如：某數除以0。
ArrayIndexOutOfBoundsException	陣列索引值超出範圍。
NegativeArraySizeException	陣列的大小為負數。
NullPointerException	物件參照為null，並使用物件成員時所產生的例外。
NumberFormatException	數值格式不符所產生的例外。

5

11-2-1 try-catch敘述

- try-catch敘述用以捕捉並處理例外

```
try{
        //例外測試區
}
catch(例外型別   例外變數){
        //例外處理區
}
```

- 只要是Throwable或其延伸類別之物件，都可以使用try-catch敘述捕捉。

6

11-2-2 多個catch區塊

try-catch敘述可以使用多個catch區塊

```
try {
        //例外測試區
}
catch(例外型別1  例外名1){
        //例外處理區1
}
catch(例外型別2  例外名2){
        //例外處理區2
}
...

catch(例外型別N  例外名N){
        //例外處理區N
}
```

7

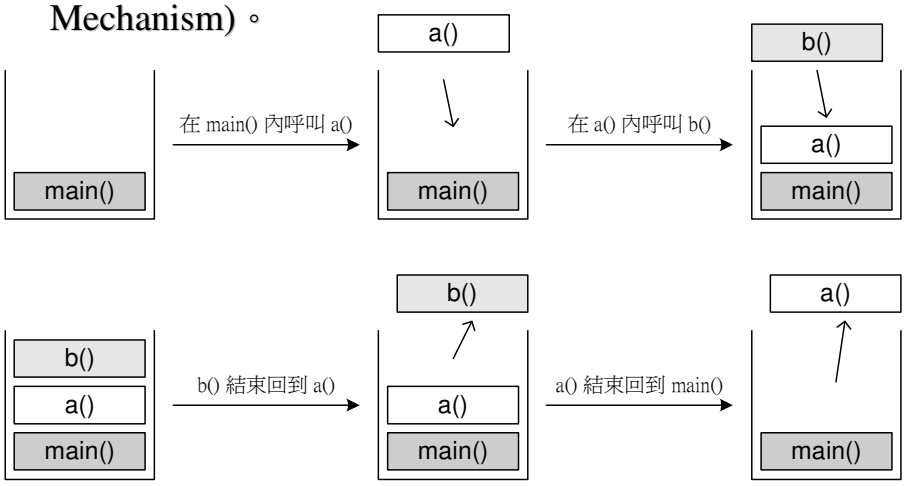
11-2-2 多個catch區塊

- 頂多只會有一個catch區塊內的敘述被執行。
- 特化的例外型別須放在較前的catch敘述，否則在編譯時就會發生錯誤。

8

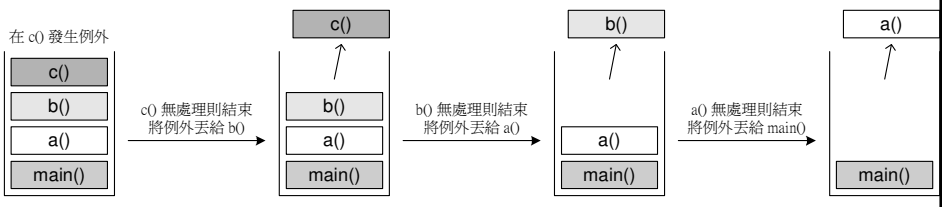
11-2-3 Call Stack機制

- 方法之間的呼叫是依循呼叫堆疊機制(Call Stack Mechanism)。



11-2-3 Call Stack機制

- 例外與呼叫堆疊機制



10

11-2-4 finally區塊

- 不論有沒有例外發生，finally區塊內的敘述皆會執行。

```
try {
        //例外測試區
}
catch(例外型別  例外名){
        //例外處理區
}
finally {
        //鐵定執行區
}
```


11-2-4 finally區塊

- finally區塊不會執行或不會完全執行：
  - 有其它例外在finally區塊中發生。
  - 在try或catch區塊使用System.exit()離開程式。
  - 未執行至finally區塊，執行緒即進入結束狀態(dead state)。

11-3-1 以throw丟出例外物件

使用throw敘述在程式中丟出例外物件

```
throw 例外物件;
```

- 丟出一包含訊息的自訂例外物件：

```
throw new RuntimeException("除數為零");
```

```
throw new Exception("找不到檔案");
```

- throw所丟出的例外物件同樣可以使用try-catch敘述處理。

13

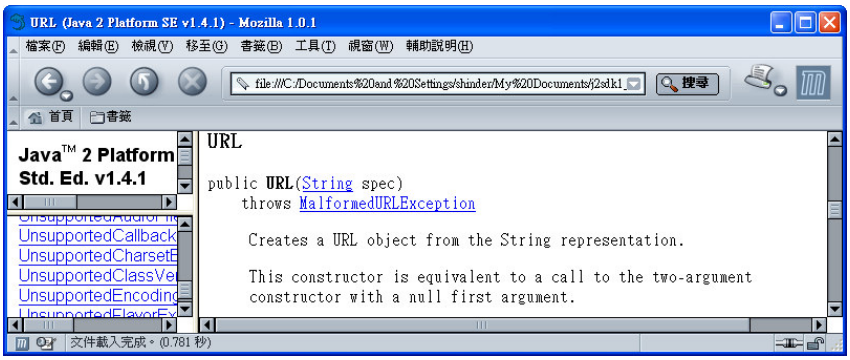
11-3-2 throws關鍵字

- **throws**關鍵字使用於方法或建構子定義的標頭，用來指出例外發生時，由方法(或建構子)丟出的例外型別。
- throws之後可以接多個例外型別名，表示方法執行過程中可能丟出屬於這些例外型別的物件。
- 方法不可以使用throw丟出不包含在throws宣告的例外型別之物件

14

11-3-2 throws 關鍵字

- URL的建構子定義使用到throws關鍵字，丟出MalformedURLException型別的例外。



15

11-3-2 throws 關鍵字

- 發生Checked Exception時須遵守**處理或宣告丟出法則(The Handle or Declare Rule)**：不是以 try-catch-finally 敘述處理就是以throws宣告丟出。

16

11-4-1 自訂例外類別

- 自訂的例外類別和JDK提供的例外類別一樣，必須為java.lang.Throwable的延伸類別。
- **printStackTrace()** 方法定義於Throwable類別內，其功能是印出呼叫堆疊的內容。

17

11-4-2 例外與方法覆蓋

- 定義覆蓋方法時，若被覆蓋的方法(overridden method)宣告丟出例外型別ExceptionA，則覆蓋方法(overriding method)宣告丟出的例外型別必須為ExceptionA或其子類別
- 所有**延伸類別**的物件都屬於**基礎類別**物件，因此，當一個延伸類別物件的方法(覆蓋父類別方法)丟出一個超出呼叫者可以處理的例外時，便會發生錯誤。

18

- 斷言(Assertion)是J2SE 1.4新增的功能，其目的是為了方便程式的除錯。
- 斷言的用處是在於**去除邏輯上的錯誤**。

19

11-5-1　斷言的宣告

第一種斷言的宣告語法：

```
assert 布林運算式;
```

- **布林運算式**的運算結果必須為**布林值**，否則在程式編譯時會發生錯誤。
- 布林運算式的結果若為false，則斷言將丟出一個沒有任何錯誤訊息的AssertionError物件。
- AssertionError類別屬於java.lang套件，為Error類別的子類別。

20

11-5-1 斷言的宣告

- 第二種斷言的宣告語法：

```
assert 布林運算式 : 訊息運算式;
```

  - 布林運算式為false時，會丟出AssertionError物件。
  - AssertionError物件所包含的錯誤訊息是由訊息運算式所構成。
  - **訊息運算式**結果通常為String物件，其運算結果不可以是void(回傳型別為void的方法)。

11-5-2 啟動斷言功能

- 斷言是J2SE 1.4新增的功能，所以程式裡包含斷言敘述時，必須使用1.4以上的編譯器版本編譯。
- 使用「-source」參數指明編譯出來的bytecode為1.4版：

```
C:\>javac –source 1.4 FileName.java
```

11-5-2 啟動斷言功能

- 執行時，使用「-enableassertions」或「-ea」參數，指明啟動斷言功能：

```
C:\>java -ea FileName
```

- 沒有使用「-enableassertions」或「-ea」啟動斷言功能，則程式中的斷言敘述會被解譯器忽略。
- 一般只有在程式測試時(軟體開發過程)，才會開啟斷言功能。

23

- 在C/C++程式中，記憶體資源被配置後若不再使用，需由程式設計者釋放。
- 在Java，記憶體資源的釋放問題由**資源回收**(Garbage Collection)機制管控。

24

11-6-1 資源回收者

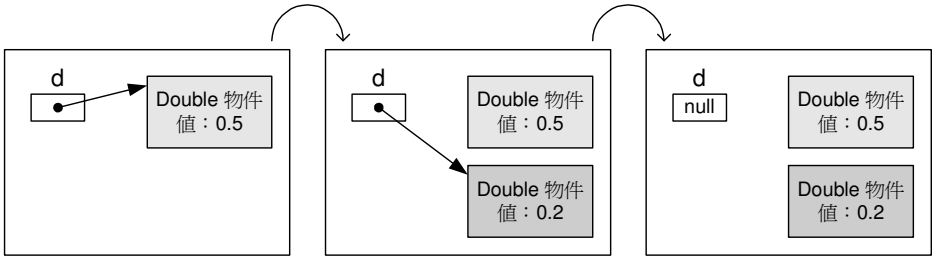
- 當物件不再被參照變數(reference variable)參考(refer)時，它就會被認為是不再使用的garbage。

```
Double d = new Double(0.5);
d = new Double(0.2);
d = null;
```

25

11-6-1 資源回收者

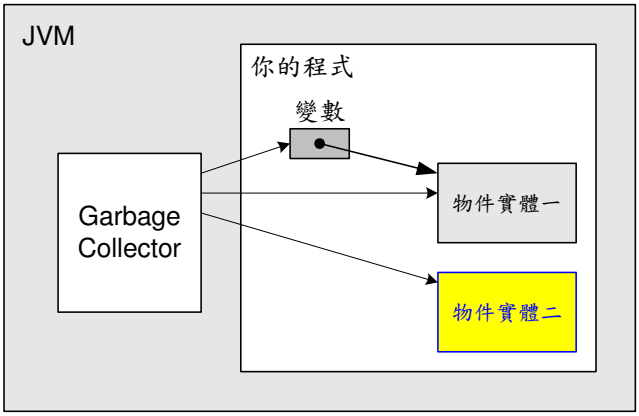
```
Double d = new Double(0.5);
d = new Double(0.2);
d = null;
```



26

11-6-1 資源回收者

- Garbage Collector在程式一開始執行時就會監視(trace)每一個變數和物件實體，當物件實體不被參考時，此物件實體即成為Unreachable Object。



27

11-6-1 資源回收者

- Unreachable Object佔用的資源並不一定馬上就會被釋放。
- 資源釋放和執行環境(可用記憶體多寡，CPU繁忙程度等)有關，程式設計者無法預測或決定資源何時被釋放。
- 執行資源回收的動作只能被建議，程式設計者無法強迫(force)資源回收的執行。

28

11-6-2 finalize()方法

- Garbage Collection機制在終結物件並釋放記憶體之前，會先呼叫物件中的finalize()方法。
- finalize()方法定義於Object類別，不過為空方法，主要是讓延伸類別覆蓋之用。

```
protected void finalize()throws Throwable
```

- finalize()方法不提供多載的功能。而且finalize()不接受傳入參數及回傳值，所以一律以void定義，並且無參數。

29

11-6-2 finalize()方法

- 建議Garbage Collector進行資源回收的方式：

```
System.gc();
```

```
Runtime.getRuntime().gc();
```

30