例外處理

TYIC 桃高資訊社

錯誤

若程式碼有寫錯，可能就會出現錯誤(error)
在編譯時發生的錯誤稱為編譯錯誤(compile error)
在執行時發生的錯誤稱為執行錯誤(runtime error)
編譯錯誤在編譯時就馬上會發生
但執行錯誤要等到真正執行時才會發生，非常的危險
大部分的執行錯誤都是例外(exception)
可以進行例外處理(exception handling)，避免程式直接中斷
少部分嚴重錯誤不是例外，不用進行例外處理，因為也處理不了
如：記憶體不足錯誤、輸入輸出串流錯誤、斷言錯誤等

例外

最常見的例外便是 java.lang.NullPointerException

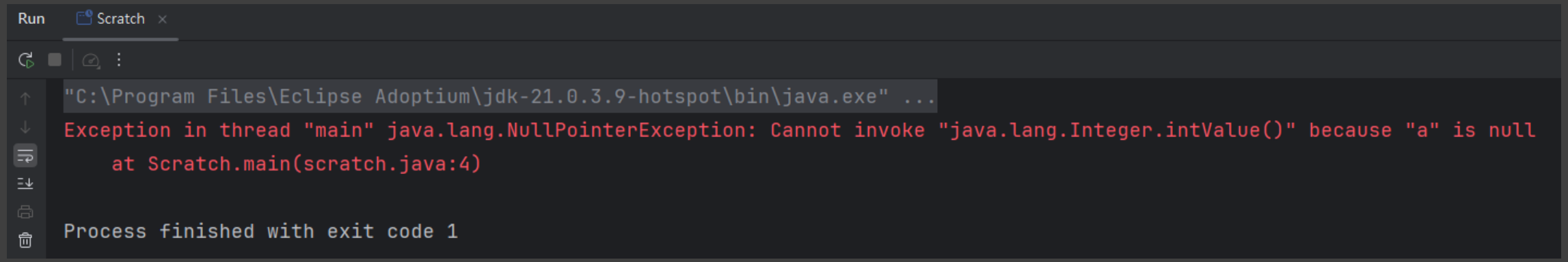
該例外會在

對 null 操作時

拋出(throw)

並在主控台顯示例外資訊

```java
class Scratch {
    public static void main(String[] args) {
        Integer a = null;
        System.out.println(a > 0);
    }
}
```

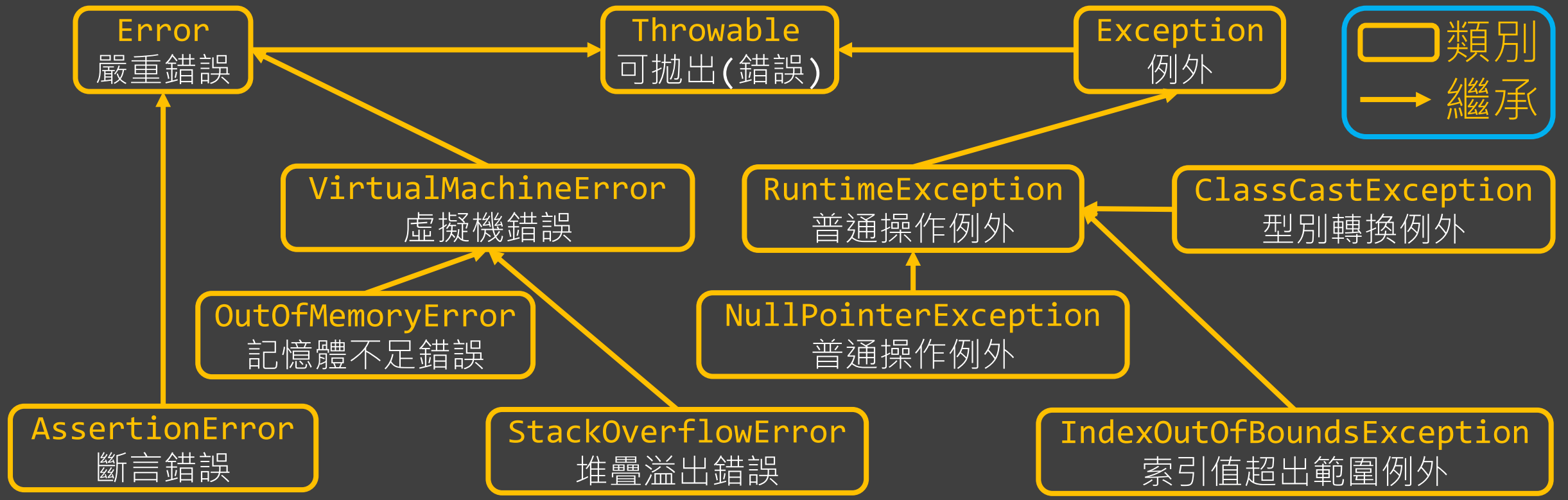


例外

每個錯誤都是一個類別，並且皆繼承 java.lang.Throwable

Throwable 有兩個直接子類別：Exception 和 Error

其中 Exception 類別代表例外，而 Error 類別代表嚴重錯誤



例外處理

在 Java 中，使用 try...catch 流程控制陳述式進行例外處理

```java
try {
    陳述式...
} catch (捕捉錯誤類別 捕捉錯誤變數名稱) {
    陳述式...
} catch (捕捉錯誤類別 捕捉錯誤變數名稱) {
    陳述式...
}
```

當 try 區塊中的程式碼拋出錯誤，且錯誤類別為捕捉錯誤類別

程式就不會終止，而是會結束 try 區塊並跳轉到 catch 區塊

且捕捉錯誤變數會初始化為捕捉(catch)到的錯誤

catch 至少需要一個，且較上方的 catch 優先於較下方的

例外處理

若捕捉多種錯誤後都要執行相同的程式碼

可以使用多重捕捉，不過須注意，多重捕捉的錯誤不可有繼承關係

```java
try {
    陳述式...
} catch (捕捉錯誤類別1 | 捕捉錯誤類別2 | ... | 捕捉錯誤類別n 捕捉錯誤變數名稱) {
    陳述式...
}
```

例外處理

```java
import java.util.Scanner;

public class Main1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String[] strings = scanner.nextLine().split(" ");
        while (scanner.hasNextInt()) {
            int index = scanner.nextInt();
            try {
                System.out.println(strings[index]);
            } catch (IndexOutOfBoundsException e) {
                System.out.println("索引值超出範圍！");
            }
        }
    }
}
```

java

```
a b c d e
1
b
5
索引值超出範圍！
4
e
^D
```

console

例外拋出

拋出例外

須使用 throw 陳述式

```
throw 錯誤物件;
```
java

```java
import java.util.Scanner;

public class Main2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String[] strings = scanner.nextLine().split(" ");
        while (scanner.hasNextInt()) {
            int index = scanner.nextInt();
            try {
                System.out.println(strings[index]);
            } catch (IndexOutOfBoundsException e) {
                System.out.println("索引值超出範圍！");
                throw e;
            }
        }
    }
}
```
java

```
a b c d e
4
e
6
索引值超出範圍！
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
    Index 6 out of bounds for length 5
        at Main2.main(Main2.java:10)
```
console

Throwable

Throwable 類別和其子類別皆有四種公開建構子：

```
Throwable()
Throwable(String message)
Throwable(Throwable cause)
Throwable(String message, Throwable cause)
```

下方為 Throwable 類別的部分公開動態方法：

```
void printStackTrace()
String getMessage()、Throwable getCause()
```

Throwable

```
a b c d e
6
錯誤訊息：索引值超出範圍
造成錯誤原因：Index 6 out of bounds for length 5
java.lang.IllegalArgumentException: 索引值超出範圍
        at Main3.printElement(Main3.java:23)
        at Main3.main(Main3.java:10)
Caused by: java.lang.ArrayIndexOutOfBoundsException:
    Index 6 out of bounds for length 5
        at Main3.printElement(Main3.java:21)
        ... 1 more
4
e
^D
```

console

```java
import java.util.Scanner;

public class Main3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String[] strings = scanner.nextLine().split(" ");
        while (scanner.hasNextInt()) {
            int index = scanner.nextInt();
            try {
                printElement(strings, index);
            } catch (IllegalArgumentException e) {
                System.out.println("錯誤訊息：" +
                        e.getMessage());
                System.out.println("造成錯誤原因：" +
                        e.getCause().getMessage());
                e.printStackTrace();
            }
        }
    }

    public static void printElement(String[] arr, int index) {
        try {
            System.out.println(arr[index]);
        } catch (ArrayIndexOutOfBoundsException e) {
            throw new IllegalArgumentException("索引值超出範圍", e);
        }
    }
}
```

java