Chapter 9 Exception Handling

1. Exception: some type of error
2. throwing an exception: mechanism that signals when something unusual happens
3. handling the exception: code that deals with the exceptional case
4. event-driven programming
    a. objects are defined so that they send events(objects) to other objects that handle the events
    b. firing an event: sending an event
5. syntax example:

```
try {
        /* 重點：
        1. 如果這個block有error，會中斷，並立即執行以下對應的catch block

        2. 如果在這裡有error，需要用if-else statement檢查是哪一種error，
           再寫throw Exception，使以下catch block接收

        3. 如果在這裡執行的method有error，而且該method可回傳throw
           Exception，就不需要寫if-else statement來throw Exception了
           ，會對應到下面Exception類別自動catch
        */
} catch (MyExceptionType myExcept) {
        //MyExceptionType為開法者自物件類別
        //myExcept為reference指向MyExceptionType物件
} catch (Exception otherExcept) {
        //Exception為Java原始物件類別
} finally {
        /* 重點：
        1. 無論有無exception，前面的block執行完，最後這裡永遠都會執行

        2. finally block在try-catch結構中可省略
        */
}
```

6. 使用try-catch重點：
    a. 不會明確知道try block哪一行出問題，因為到底哪一行出問題不重要
    b. 若在try block呼叫的method可回傳throw Exception，就可以省去在try block內寫一大堆if-else的檢查機制
    c. catch block只有遇到錯誤才會直行。執行完後會立即執行finally block

d. `finally` block可省略，並非必要的。`finally` block無論有無錯誤都會執行

7. ArrayIndexOutOfBoundsException example

```
public class HelloException {
      public static void main(String[] args) {
            int i=0;
            String greetings[] = {
                        "Hello world!",
                        "hello",
                        "world",
            };

            while(i<4){
                  try{
                        //不需要寫if-else，超出範圍直接catch
                        System.out.println(greetings[i]);
                        i++;
                  } catch (ArrayIndexOutOfBoundsException e){
                        System.out.println("catch");
                        break;
                  } finally{
                        //無論有無錯誤都會執行
                        System.out.println("Always printed");
                  }
            }
      }
}
```

8. self-defined example

```
//derived from Exception
public class DividedByZero extends Exception {
	public DividedByZero(){}

	public DividedByZero(String message){
		super(message);
	}
}

public class Test {

	//func會可回傳throw DividedByZero Exception，所以try-catch可
	//以不用寫if-else
	public static int func(int x, int y) throws DividedByZero{
		if(y == 0){
			throw new DividedByZero("divided by 0");
		}
		return x/y;
	}

	public static void main(String[] args) {
		int a = 0;
		int b = 0;

		//真正要用try catch應該在這裡，而非在func內部
		//func會throw exception，如果在這裡不寫try-catch會有
		//warning
		try{
			func(a,b);
		} catch (DividedByZero e){
			//Exception class有data member存取error message
			System.out.println(e.getMessage());
		}
	}
}
```

9. Exception Controlled Loops，例如：網路連線會不斷地試圖連線

```
boolean done = false;
while (! done) {
      try {
            CodeThatMayThrowAnException
            done = true;//execute only when there is no exception
      } catch (SomeExceptionClass e){
            SomeMoreCode
      }
}
```

10. Exception Classes from Standard Packages(`java.lang` package)
    a. for example: `Exception`, `IOException`, `NoSuchMethodException`, `FileNotFoundException`
        i. `Exception` has `getMessage` method
            1. there is a String arguement in the `Exception` constructor
    b. more info: http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html
11. Other Exception Classes should be imported from `java.io.IOException`
12. multiple `catch` blocks
    a. when catching multiple exceptions, catch the more specific exception first
    b. when an exception is thrown in a try block, the catch blocks are examined in order, and the first one that matches the type is executed
    c. for example:

```
public static int func(int x, int y){
      boolean fail = true;
      try{
            if (y == 0)
                  throw new MyException("divided by 0");
            if (fail == true)
                  throw new MyException2("fail boolean");
      } catch (MyException e){
            System.out.println(e.getMessage());
            System.exit(1);
      } catch (MyException2 e){
            System.out.println(e.getMessage());
      } catch (Exception e){//越general的exception寫越後面，
                            //以免會接收了所有Exception的繼承者
            System.out.println(e.getMessage());
      }
      return x/y;
}
```

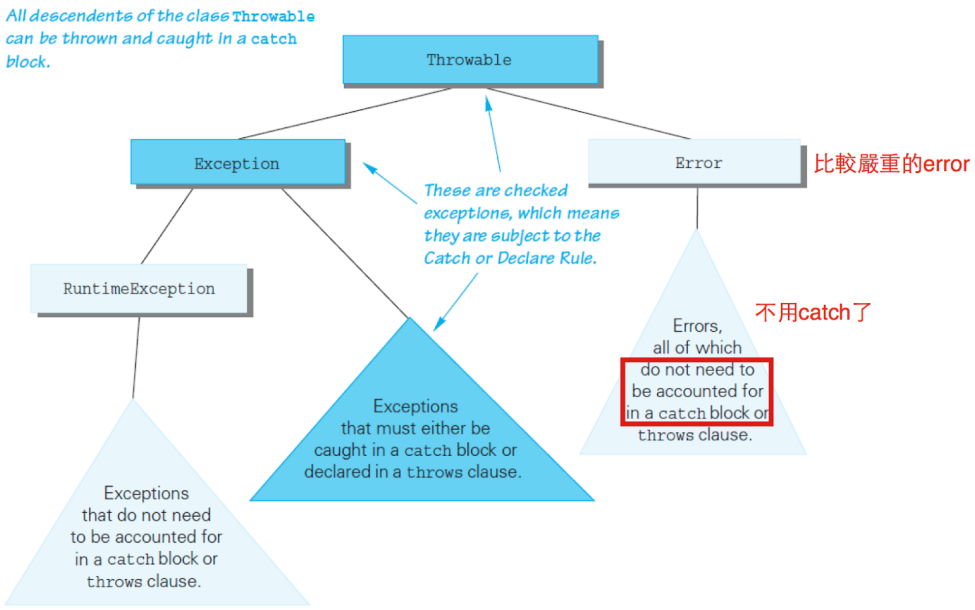
13. throwing an exception in a method
    a. 請見重點9
    b. if a method can throw more than one type of exception, separate the exception types by commas, for example:
       `public void aMethod() throws AnException, AnotherException`
    c. 以上`AnException`和`AnotherException`具備covariant特性，故可以回傳其derived class

14. Catch or Declare Rule
    a. only two techniques to catch
        i. `try` block throws an exception, and `catch` block catches the possible exception within the same method
        ii. possible exception is declared at the start of the method definition, and returns a “throw exception” if error occurs (throwing an exception in a method)
    b. both techniques can be mixed
15. checked exception: errors that occur in a correct program
    a. `Exception`, `Throwable`, and all descendants of `Exception` deals with such errors
    b. must use Catch or Declare Rule
16. unchecked exceptions: fatal errors
    a. fatal situations: `Error` and its descendant classes deals with such errors
    b. not subject to the Catch or Declare Rule
    c. case example: memory space not enough

Hierarchy of Throwable Objects

not interface

Display 9.11 Hierarchy of `Throwable` Objects



17. Nested `try-catch` blocks
    a. `try-catch` block in `catch` block
        i. must use different names for `catch` block parameters in inner and outer blocks
    b. `try-catch` block in `try` block
        i. if an exception is not caught in the inner block, then it will be thrown to the outer block
18. Rethrowing an Exception
    a. `catch` block has code that throws an exception
    b. sometimes catch an exception, and check `getMessage`, and then thrown to somewhere else
19. main method可以加上throw
    a. for example:
       `public static void main(String[] args)` **`throws Exception`**
    b. compile的時候不會出錯，但是runtime時候會出錯