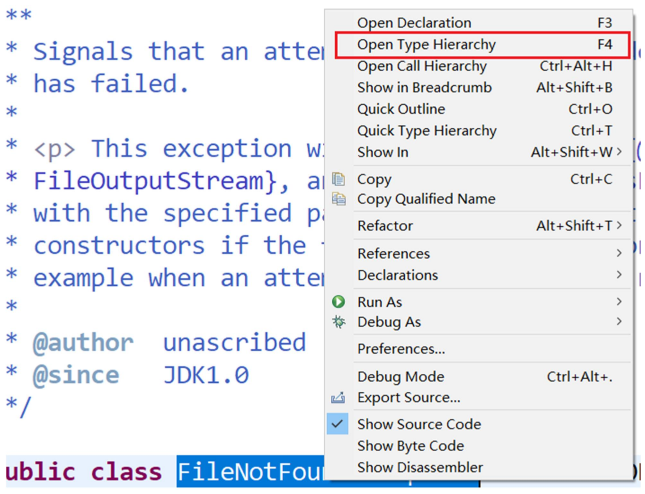
Java2 Lab 2 (Computer system design and application)

[Experimental Objective]

1. Learn to check the java class hierarchy.
2. Learn to process exceptions using try-catch.
3. Learn to throw an exception, learn something about chained exceptions.
4. Learn to trace exceptions through the call stack.
5. Learn finally block.

[Java class hierarchy]

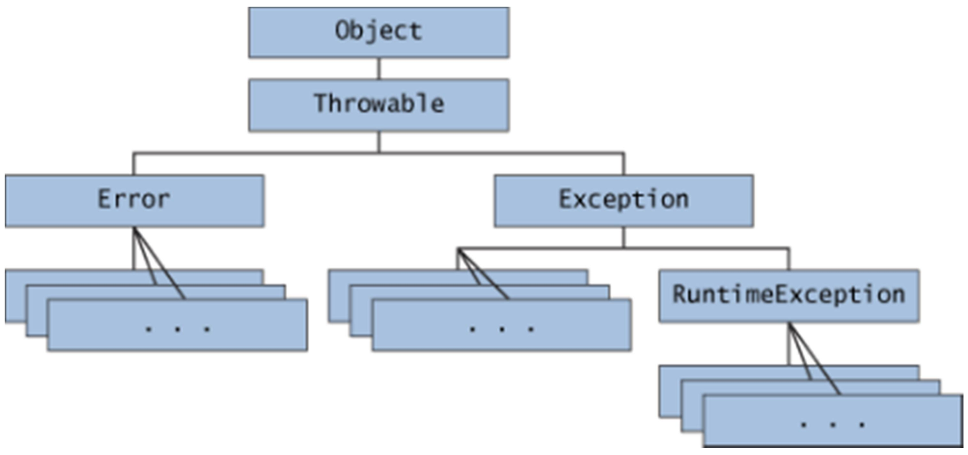
Right-click the name of the exception class, select the open type hierarchy in the popup menu.



```
**
* Signals that an atte
* has failed.
*
* <p> This exception w
* FileOutputStream}, a
* with the specified p
* constructors if the
* example when an atte
*
* @author  unascribed
* @since   JDK1.0
*/

ublic class FileNotFou
```

In java, all classes are inherited from the Object class. Error class and Exception class both are inherited from Throwable class. RuntimeExceptioin is the subclass of Exception. Try-catch block mainly used on checked exception. If a RuntimeException/Error has been thrown, try-catch block is not mandatory required.



[Try-Catch]

Tips:

We should put connected code in one try block, not separate them in different blocks.

The superclass can handle the exception of its subclass type. We should put the catch block with the subclass type before the superclass type.

Try to add try-catch to the following code to make it work.

Source code 1:

```java
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Lab2 {
	public static void main(String[] args){
		Date date = readDate();
		System.out.println("Date    = " + date);
	}

	public static Date readDate() {
		FileInputStream readfile = null;
		InputStreamReader ir = null;
		BufferedReader in = null;
		readfile = new FileInputStream("readme.txt");
		ir = new InputStreamReader(readfile);
		in = new BufferedReader(ir);
		// read one line from the file
		String str = in.readLine();
		if (str == null) {
			in.close();
			return null;
		}

		DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
		Date date = df.parse(str);
		in.close();
		return date;
	}
}
```

Tips:

`FileInputStream`: obtains input bytes from a file in a file system.

`InputStreamReader`: a bridge from byte streams to character streams.

`BufferedReader:` reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

Attention:

Notice how the inheritance of the exception-class affect the execution of the catch block.

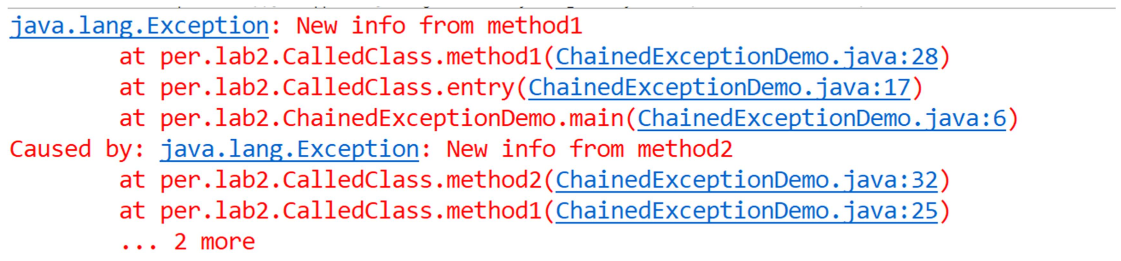
[Throws and chained exception]

Source code 2:

```java
public class ChainedExceptionDemo {
	public static void main(String[]args){
		//chained exception
		CalledClass.entry(args);
	}
}

//chained exception
class CalledClass{
	public static void entry(String[]args){
		try{
			method1();
		}
		catch(Exception ex){
			ex.printStackTrace();
		}
	}
	public static void method1()throws Exception{
		try{
			method2();
		}
		catch(Exception ex){
			throw new Exception("New info from method1",ex);
		}
	}
	public static void method2() throws Exception {
		throw new Exception("New info from method2");
	}
}
```

Output:



```
java.lang.Exception: New info from method1
        at per.lab2.CalledClass.method1(ChainedExceptionDemo.java:28)
        at per.lab2.CalledClass.entry(ChainedExceptionDemo.java:17)
        at per.lab2.ChainedExceptionDemo.main(ChainedExceptionDemo.java:6)
Caused by: java.lang.Exception: New info from method2
        at per.lab2.CalledClass.method2(ChainedExceptionDemo.java:32)
        at per.lab2.CalledClass.method1(ChainedExceptionDemo.java:25)
        ... 2 more
```

[Finally block]

Run the following code, understand the usage of finally block.

Source code 3:

```
public class AboutFinally {
    public static void main(String[] arg) {
        System.out.print(Method());
    }

    private static boolean Method() {
        try
        {
            return true;
        }
        finally
        {
            return false;
        }
    }
}
```

[Assignment] **deadline: 11:55pm, September 26.**

1. The Integer.parseInt() method requires a String arguments, but fails if the String cannot be converted to an integer. Write an application in which you try to parse a String that does not represent an integer value. Catch the NumberFormatException that is thrown, and then display an appropriate error message. Save the file as TrytoParseString.java.

2. Try to add a method named method3 to the source code 2, it is called by method2(), and throw an exception with message "New info from method3". Observe the message in the console window, save the output in the console window as a jpg named Ouput.jpg.

3. Answer the question: Under what circumstances will a finally block not execute or not execute completely? Save your answer in a txt file named Finally.txt.