**OOP Lab 9**

|  |  |
| --- | --- |
| Due Date: | **November 8, 23 : 59** |

* **Submit your assignment using the following file format:**

LabNumber\_RoomNumber\_StudentName\_Student\_ID.zip

Example: Lab9\_328\_Hongkildong\_201620505.zip

* This zip file will contain **two types of** files, namely:

1. **report file** with file format **“Report\_Lab number**” (eg. report\_9) to answer theory questions and to write the screen shot of your program.
2. Source code file that contains codes of classes to answer programming questions.

**I. Objectives**

1. Learning the usage of Exception.

2. Learning the difference between **Checked Exception** and **Unchecked Exception**.

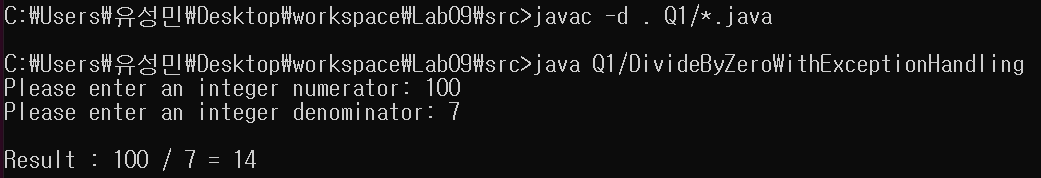
3. Learning the structure of **Exception** and **terminatio**n models.

**II. Exercises (15 points)**

**1. Answer the following questions after executing the code in Fig 11.3**

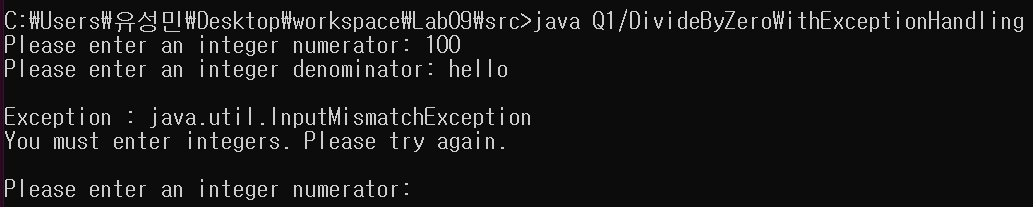
* 1. At which line number the “ **InputMismatchException** ” occur? (**1 point**)

**=>**

**When input 100 and 7, There is no error. There is no compile error at the code. The ‘inputMismatchException’ is occurred at runtime so compiler doesn’t know at which line number the ‘inputMismatchException’ occurred.**

* 1. Why the “**InputMismatchException**” occurs? (1 point)

**=>**

** Because ‘hello’ in not integer type. If you don’t want to occur error, then you must enter two integer respectively denominator and numerator.**

* 1. At which line number the “**ArithmeticException”** occurs? (1 point)

**=>**

**When we input 0 to denominator value. At the given ppt the line number is 27 where we input 0 with scanner function. And the error occurs at runtime.**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

* 1. Why the “**ArithmeticException**” occurs. (1 point)

**=>**

**Because the number can’t be divided with 0. And as we can see the java API, “ArithmeticException class is thrown when an exceptional arithmetic condition has occurred. For example, an integer “divided by zero” throws an instance of this class. ArithmeticException objects may be constructed by the virtual machine as if suppression were disabled and/or the stack trace was not writtable”.**

**2. Answer the following questions about the program in Fig 11.3**

A .Before line 34, **add** the following catch block and compile the program. Explain what happens and explain why this happens (including only explanation, **2 points**)

|  |
| --- |
| catch (**RuntimeException** re)  {  System.err.printf("%n Exception: %s%n", re) ;  } |

**=>**

**When run the code, as we can see, the RuntimeException class is super class of those InputMismactchException and ArithmeticException classes. Because the catch block of super class is declared above the child’s one, the exceptions occurred at try block will be handled at first catch block. So the other subclass’s catch block is no more used.**

텍스트이(가) 표시된 사진

자동 생성된 설명

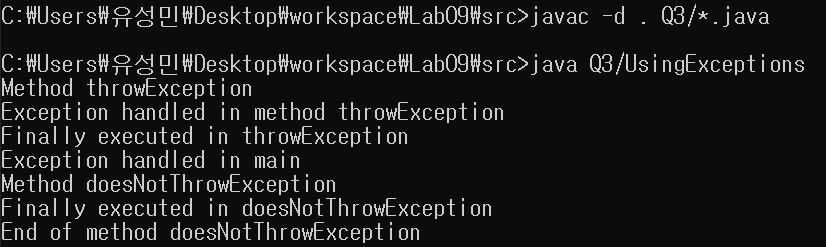
B. Replace the **second** catch block (Fig.11.3) by the following catch block. Then **enter** input similar values as shown in the example and execute the program. **Explain** what result you observed and explain why such type of results occur (including the **reason** and **screen shot of the result , 2 points**)

|  |
| --- |
| **Catch** (RuntimeException re)  {  System.err.printf("%nException: %s%n", re);  } |

**=>**

텍스트이(가) 표시된 사진

자동 생성된 설명**You can see the difference between two results. First screen shot is when changed second catch block, and second one is before changing the code. Because this type of error is not ‘InputMismatchException’ error, compiler check second catch block. And because ArithmeticException is subclass of the RuntimeException class, the error can be handled by RuntimeException catch block. But the subclass is more specifically handle the error with user defined messages. So that is the difference when handled by ArithmeticException block and RuntimeException block at this code.**

3. After deleting “**throws Exception**” at line 21(Fig 11.5), compile the program. Explain why a problem happens (**including only explanation, 2 points**)

**=>**

**At the throwException() method, there is catch block of ‘Exception e’. The method should handle the error occurred by catch block or throws exceptions. But by deleting the line 21, the method does not handle the error. So the above error occurs.**

4. After **running**  the following code, explain the reason why you got this result (including your reason and screen shot of the result, **3 points**)

|  |
| --- |
| // **A.java**  1. public class **A**  2. {  3. public static void **main**(String[] args)  4. {  5. try  6. {  7. System.out.println("A");  8. **m();**  9 . System.out.println("B");  10 }  11. catch(Exception e)  12 {  13. System.out.println("C");  14. }  15. finally  16. {  17. System.out.println("D");  18. }  19. }// end of main() method  21 public static void **m**()  22. {  23. System.out.println("E");  24. if(true)  25. **throw new RuntimeException**();  26. System.out.println("F");  27 }  28 }**// end of class** |

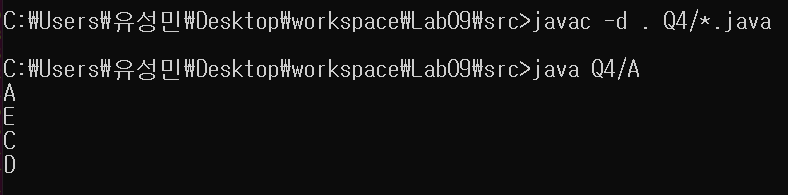
**=>**

**First at the try block we print ‘A’, then call method m().**

**Second method m() print ‘E’ then throw ‘RuntimeException’ error, so the rest code of method m() dosen’t executed and go to catch block of the main method.**

**Third by the catch block ‘C’ is printed.**

**Fourth ‘D’ of the finally block is executed. ‘Finally block’ is executed after catch block.**

****

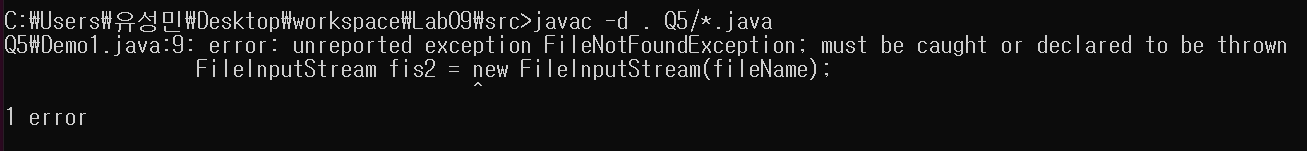
**5. When you compile the following program, an error occurs.**

|  |
| --- |
| **import** java.io.\*;  **class** Demo1  {  **public** **static** FileInputStream **CreateFile**(String **fileName**)  {  FileInputStream **fis2** = **new** FileInputStream(**fileName**);  System.*out*.println("File input stream is created");  **return** **fis2**;  }  **public** **static** **void** main(String args[])  {  **FileInputStream fis1 = null;**  String **fileName** = "foo.bar";  System.*out*.println("File name is " **+** fileName);  Fis1 = ***CreateFile***(**fileName**);  System.*out*.println("End of the program");  }  **}** |

A. Explain **why** the error happens.

**=>**

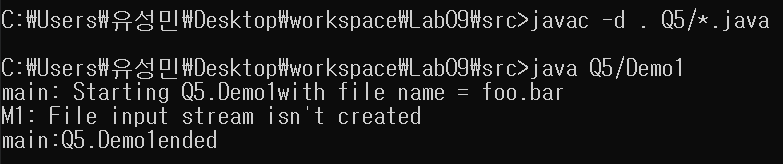
**The ‘FileNotFoundException’ was not handled at two methods. So errors occur. We can handle the error by adding throws exception or handling at method block by try catch block.**



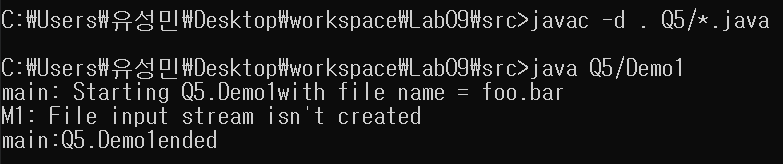
B. In order to do **normal** operation, **add** the exception handling code using two ways. In your report, include screen shot of errors, explanation for the occurrence of the error, screen shot of the normal code (without error). Include also the source code of the normal code in the source file (**4 points**)

=>

**First way is making try-catch block at ‘M1()’ method to handle error in the method M1(String fileName).**

****

**Second way is throwing exception at M1() method then handle at main method.**



**The above answers to the two ways are contained in the source code and commented.**