CST 8284 Object Oriented Programming (JAVA)

ASSIGNMENT 2

Instructions

You are required to complete and submitted this assignment by yourself individually. You must ensure that your work is NOT **replicated** by any other student, and that you do not copy or submit someone else's work. You are equally responsible if your code shows up into someone's code. These scenarios could constitute plagiarism or academic dishonesty.

The **due date** for submission of this assignment is **Sunday, November 27 by 11:59PM. The demo due date for this assignment is the week of November 28 (whichever day your lab sessions falls into that week).** Read every instruction and the tasks carefully to ensure that you complete all aspects of the solution required. **You will need to demo this** assignment during your lab session.

Scenario

Exception handling is an important aspect of programming that provides a uniform technique for detecting, documenting, and recovering from errors. It facilitates Programmers' understanding of each other's error processing code (for example when working on large projects).

While designing your systems, adding exception handling and error recovery techniques is important from the beginning. This is because that doing so after a system has been implemented can be very difficult and/or costly.

In this assignment, you will explore exception handling for different scenarios.

Java exception classes inherit directly or indirectly from class Exception (see **Throwable** inheritance hierarchy - the superclass of class Exception (part of the hierarchy of exception classes) - figure shown below in this document.

There are **four parts** to this assignment, which cumulatively carry a weight of 9% of your total course mark.

SECTION ONE

1. Leveraging your knowledge of relevant concepts in OOP (Java):

    a. Create **CatHandler.java** which contains the **main** method.
    b. Inside **CatHandler**, declare an inner class named ExceptionAlpha that extends Exception.
    c. Add two exception **subclasses** named ExceptionBeta and ExceptionGammer, where **ExceptionGammer** extends from ExceptionBeta, and ExceptionBeta extends from ExceptionAlpha. ExceptionBeta and ExceptionGammer are also inner classes.

2. In **CatHandler.java**, write code that shows that the subclass exceptions ExceptionBeta and ExceptionGammer will be caught in the **catch block** of type ExceptionAlpha. (Refer to the notes and course text). These three exception subclasses are **empty**; they contain no code.

3. For the output, one can use **System.err.println(), getMessage()** and **printStackTrace() and other appropriate print statements** to show that the exception subclasses have been successfully caught. You are required to select the right choice(s) of these method in each case to show that the exception subclasses have been successfully caught.

Hint: Check to see **Throwable** Class Hierarchy in the diagram provided (for sub- and super classes).

SECTION TWO

In this part of the assignment, you are required to write a program which demonstrates the use of the catch block using catch (Exception exception).

1. To proceed, define the following classes:
   a. Create a new test class called **DogHandler.java** which contains your main method.
   b. Add inner class ExceptionDog that **inherits** from Exception
   c. Add inner class ExceptionPuppy that **inherits** from class ExceptionDog
   d. These exception subclasses are **empty**; they contain no code.

2. Your program must have **try catch blocks** which throw exceptions of types:
   a. ExceptionDog
   b. ExceptionPuppy
   c. NullPointerException
   d. IOException

3. Ensure that all exceptions are caught by your test method **DogHandler.java** where the catch block is **parameterized** with type Exception **in four separate methods.**

4. For the output, one can use **System.err.println(), getMessage()** and **printStackTrace() and other appropriate print statements** to show that the exception subclasses have been successfully caught**.** You are required to select the right choice(s) of these method in each case to show that the exception subclasses have been successfully caught.

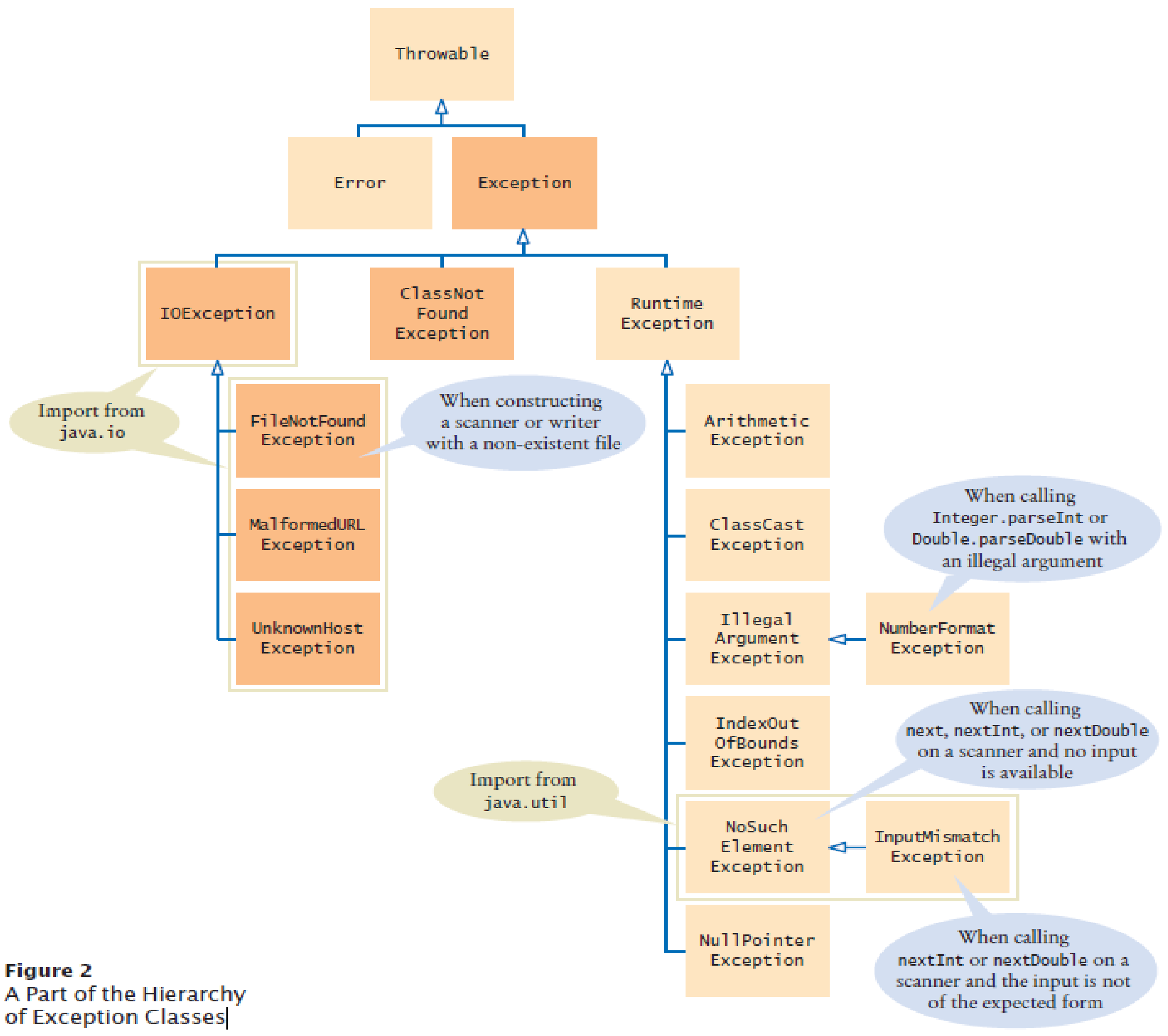
Throwable
Error
Exception
IOException
ClassNot Found Exception
Runtime Exception
Import from java.io
FileNotFound Exception
When constructing a scanner or writer with a non-existent file
MalformedURL Exception
UnknownHost Exception
Arithmetic Exception
ClassCast Exception
When calling Integer.parseInt or Double.parseDouble with an illegal argument
Illegal Argument Exception
NumberFormat Exception
IndexOut OfBounds Exception
When calling next, nextInt, or nextDouble on a scanner and no input is available
Import from java.util
NoSuch Element Exception
InputMismatch Exception
NullPointer Exception
When calling nextInt or nextDouble on a scanner and the input is not of the expected form

Figure 2
A Part of the Hierarchy of Exception Classes

SECTION THREE

In this part, you are required to demonstrate **rethrowing** of an exception.

1. Write **FishHandler.java** that demonstrates the **rethrowing** of an exception:
   a. Define **two methods** in **FishHandler.java** named:
      I. easterEnding(), which should initially throw an Exception.
      II. easterStarting(), which calls easterEnding, and catches the Exception and rethrows it.

b. From your main method, initiate a call to easterStarting, and catch the rethrown exception.

2. For the output, one can use **System.err.println(), getMessage()** and **printStackTrace() and other appropriate print statements** to show that the exception subclasses have been successfully caught**.** You are required to select the right choice(s) of these method in each case to show that the exception subclasses have been successfully caught.

SECTION FOUR

In this part of the assignment, you are required to show that the **arrangement (order)** of the catch blocks in your program is essential.

1. In your program, create a class named **OrderHandler.java**
2. In main method, instantiate a new **superclass** exception type Exception
3. In main method, instantiate a new **subclass** exception type IOException
4. **OrderHandler.java** should show a compilation error when you try catching the superclass exception type **before** the subclass exception type.
5. Rewrite your code in solution (4) above to show no compilation error.

Important Information

1. **Each** of the four parts should demonstrate the requirements specified for that particular part.
2. For all aspects of your programs you must provide appropriate **Javadoc** style comments and documentation. You are required to submit a generated Javadoc folder.
3. D**emonstrate** your code output by submitting **screenshots** of your **eclipse screen** for each successful run of the four parts. **Text** output files will NOT be accepted.
4. You are required to submit **four Java code files (one for each part)** and screen shots for each section. Your screen shots must show the last three lines of your code on eclipse to the entire output.
5. Zip all your project containing the four java files and submit to your **Lab Section Assignment 2** drop box using the file name format: **YourFirstName_YourLastName_ Section_XXX**. Ensure that you follow any lab specific instructions for submission that your professor may provide.