Exercises on Exceptions

1. Write a java program using multiple catch blocks. Create a class ExceptionExercise inside the try block declare an array a[] with size 5 and initialize with value a[5] =30/5; . In each catch block show Arithmetic exception and ArrayIndexOutOfBoundsException.

***Test Data:***

a[5] =30/5;

***Expected Output :***

ArrayIndexOutOfBoundsException occursRest of the code

**Solution:**

import java.lang.\*;  
import java.io.\*;  
  
public class CatchExercise {  
 public static void main(String[] args) {  
  
   try {  
  
     int a[] = new int[5];  
     a[5] = 30 / 5;  
   } catch (ArithmeticException e) {  
  
     System.out.println("ArithmeticException occurs");  
   } catch (ArrayIndexOutOfBoundsException e) {  
  
     System.out.println(" ArrayIndexOutOfBoundsException occurs");  
   } catch (Exception e) {  
     System.out.println("Parent Exception occurs");  
   }  
   System.out.println("Rest of the code");  
 }  
}  
  
/\*OUTPUT:  
ArrayIndexOutOfBoundsException occurs  
Rest of the code  \*/

**Explanation:**

In class CatchExercise inside the try block declare an array a[] and initialize with value a[5] =30/5; . In each catch block show Arithmetic exception and ArrayIndexOutOfBoundsException with the exception object e. Hence the array size is 5 and index starts from 0. It does'nt hold the index 5 value so it shows ArrayIndexOutOfBoundsException.

1. The professor you TA (Training Associate) for, Professor Jackson, shared with you the code she uses to auto-grade students’ work. She and the other TAs have encountered some problems with the code in the past when they enter the total possible point value for an assignment. Occasionally, they accidentally enter **0** for the total number of possible points and the program encounters a fatal error when trying to divide by 0.

To help out with this issue, complete a function called **divide()** in **Main**.

The **Divide()** method takes in two parameters: **x** and **y**.

Your function should return the result of **x/y**.

However, if **y** is zero, you should throw an exception.

Try to use an **ArithmeticException** and put your **try/catch** block in **divide()** to test out your error-handling skills.

If an exception is caught, make sure to print out a helpful message.

1. After mentioning to Professor Jackson that you would like to get some more practice with exceptions, she offered to let you write some grading software! Before she gives you full control over auto-grading students’ work, she asked if you could write a function called **CheckFileExtension()**.

The **CheckFileExtension()** method takes in one parameter: **fileName**.

**CheckFileExtension()** should return an integer representing the number of points a student receives for properly submitting a file in Java.

If a student’s submitted file ends in **.java**, they get 1 point.

If a student’s submitted file doesn’t end in **.java**, they get 0 points.

If the file submitted is **null** or an empty string, an exception should be thrown and you should give the student -1 points.

What kind of exception is up to you, including to a custom exception!

In **main()**, Professor Jackson has provided a array of students and the names of their submitted files for you to test out your work.

If an exception is caught, make sure to print out the error message.

1. Take a look at the **Temperature** class

**public** **class** Temperature {

**private** **double** fahrenheit;

**public** **double** **getFahrenheit**() {

**return** fahrenheit;

}

**public** **void** **setFahrenheit**(**double** aFahrenheit) {

**double** absoluteZeroFahrenheit = -**459.67**;

**if** (aFahrenheit < absoluteZeroFahrenheit) {

System.**out**.**println**("Value is below absolute zero");

}

fahrenheit = aFahrenheit;

}

}

Instead of simply printing out a warning that the temperature is below absolute zero, let’s use exceptions to handle this case.

Before coding with exceptions, first think about what the exception needs to handle. In the case of our **Temperature** class, we need to throw an exception if the program attempts to set the value of **fahrenheit** to a temperature that is below absolute zero.

The first question we need to ask about this exception is: when will it be thrown? Exceptions in Java can either be checked or unchecked depending on if the exception is a compile-time exception or a runtime exception.

In the context of a larger program, **setFahrenheit()** is only called with a value that is given by a user. This means if we throw an exception when a value that is below absolute zero is passed to **setFahrenheit()**, it will be a runtime or unchecked exception.

Now that we know we are going to use a runtime exception, we need to answer the question: under what conditions should the exception be thrown?

In our case, that is when we attempt to set **fahrenheit** to a value that is below -459.67. Based on our knowledge of builtin Java exceptions, we know that there isn’t a builtin exception that works for this, so now we need to build a custom exception called the **TemperatureException** class.

**So write TemperatureException class**

We need to use this exception to prevent the temperature from being set to a value that is below absolute zero

1. **Do the following**:

Create a class Book with id, name as the member variables.

Create BookNotFoundException as a checked exception and create one argument constructor which is accepting String message.

Create a test class with an array of 5 books with ids starting from 1.

Create a method findById(Book[] books , int id) which will take the previously created book array and one id to find the book.

Print "Found" if the book is present with the given id else throw BookNotFoundException with a message "Book with the given id is not present".

Handle the exception in main method and print the stacktrace in catch block.