**Student Name: Tolegenov Bekzat**

Please, send it to pakita.kbtu@gmail.com

1. **A Simple Example**:

This part of the lab considers a simple example of exception handling.

1. Open Example1.java
2. Compile and execute the application Example1.

**What was output by the application when you executed it?**

The answer is: 2

Done.

1. Change the value of denominator to 0.
2. Re-compile and re-execute Example1.

**What "error" was generated by the application when you executed it?**

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Task1.Example1.main(Example1.java:12)

**because zero cannot be a denominator and it is not arithmetically properly to divide by zero**

**Why was this "error" generated at run-time (rather than at compile-time)?**

**Because the code is constructed correctly (has correct syntax) and there will be an exception only while running time**

1. Add a try-catch statement. Specifically, put only the statement that generated the exception inside of the try block and put no statements in the catch block. (Hint: You should be able to determine what exception to catch and what line generated the exception from the error message that you received during the previous step.)

Re-compile Example1.

**What error is generated and why?**Noting, the exception was catch and in there is no any statement catch block

1. Move the "output statement" into the try block (as well).
2. Add the statement System.out.println("Divide by 0."); to the catch block.

Re-compile and re-execute Example1.

**What output was generated?**

8. Add a call to the printStackTrace() method of the ArithmeticException to the end of the catch block.

Re-compile and re-execute Example1.

**What output was generated?**

**Did the application execute properly or not?**

1. **A More Complicated Example:**

This part of the lab considers an example of exception handling within and outside of block statements.

* Open Example2.java
* Compile Example2.

**What error was generated?**

* Initialize i to 0 inside of the try block (but before the for loop).

* Compile Example2.

**What error was generated?**

* It is not possible for i to be used before it is initialized. Why is this error generated anyway? (Hint: Think about block statements.)
* Move the initialization of i before the try block.

* Compile and execute Example2.

**What output is generated?**

**Why aren't all of the divisions even attempted?**

* Fix Example2 so that it executes properly. (Hint: Move the try-catch block inside of the

for block.) What did you change? What has happened?

1. **An Inappropriate Use of Exception Handling**

This part of the lab considers an inappropriate use of exception handling and how to "fix" it.

* Compile and execute Example3 and verify that it outputs all of the values followed by the word "Done".
* Modify Example3 so that it loops "properly" and does not need to use a try-catch statement. (Note: The output should not change.) **What did you change?**

1. **Some Other Exceptions**

This part of the lab will give you some experience with some other exceptions, where they arise, and how they can be used.

* **What functionality does a StringTokenizer object provide? Give example.**
* **What are the three formal parameters of the explicit value constructor in the StringTokenizer class? Give example.**
* Run Example4.java.

* After running the program write the following to the command line: 5.3+9.2

**What output is generated?**

* Now run it again and enter the following: 5.3+

**What output is generated?**

**Why? In particular, what exception is thrown and why?**

* Run again and enter 5.3+a.

**What output is generated?**

**Why? In particular, what exception is thrown and why?**

**5. Programming Practice**

* Modify Example4.java so that it supports addition (+), subtraction (-), multiplication (\*), and division (/).

* Modify Example4.java so that it processes more than just one expression for evaluation. So, for example, it should be able to be execute the following input:

4 5.0+4.1 3.2\*9.1.

* Modify Example4.java so that it tells you which operand is not a number. (Hint: You may need to use nested try-catch blocks.)

**My programs (modified):** (paste directly from eclipse)

**Example1**

package Task1;

public class Example1

{

public static void main(String[] args)

{

int denominator, numerator, ratio;

numerator = 5;

denominator = 0;

try {

ratio = numerator / denominator;

System.out.println("The answer is: "+ratio);

System.out.println("Done."); // Don't move this line

}

catch(ArithmeticException ae)

{

System.out.println("Divide by 0.");

}

}

}

**Example2**

**Example3**

**Example4**