**Student Name: Amangeldi Islam**

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?**

**2**

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?**

**ArithmeticException error, when I tried to divide by 0**

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

**Because we tried to divide by 0**

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?** Also divide by zero

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?**

**Divide by 0**

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?**

**Divide by 0, and below error ArithmException**

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

**After using try, catch yes, before no**

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?**

**Compilation error of int i;**

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

* Compile Example2.

**What error was generated?**

**Compilation error of i;**

* 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?**

100/10=10

Couldn't calculate 10/0

**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?

We obtained all of divisions, even it is over zero.

100/10=10

Couldn't calculate 10/0

0/5=0

5/2=2

2/8=0

Couldn't calculate 8/0

0/30=0

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?**

I just changed range of [i, to size of array] without using try, catch block.

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.**

**Divides string into parts, example of this is: “I just done reading book”, output is “I”, “just”, “done”, “reading”, “book”. So, it divides into parts< where between is a space.**

* **What are the three formal parameters of the explicit value constructor in the StringTokenizer class? Give example.**

**String token = new StringTokenizer(String).**

**String token = new StringTokenizer(String, String).**

**String token = new StringTokenizer(String, String, boolean).**

**String token = new StringTokenizer(“I just came”);-String that is going to be parsed**

**String token = new StringTokenizer(“I just came”, “ “) – First is string that is going to be parsed, Second string is delimeter, that we divide by).**

**String token = new String Tokenizer(“I just came”, “ “, true) - First is string that is going to be parsed, Second string is delimeter, that we divide by, Third Boolean finds out whether it is skipped delimeter (if false) or also token (if true).**

* Run Example4.java.

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

**What output is generated?**

**Output is 14.5**

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

**What output is generated?**

**Invalid syntax**

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

**NoSuchElementException is thrown, because rightOperand is null.**

* Run again and enter 5.3+a.

**What output is generated?**

One or more operands is not a number

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

**NumberFormatException, because rightOperand is not a number.**

**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**

**import** java.lang.\*;

**public** **class** Example1

{

**public** **static** **void** main(String[] args)

{

**int** denominator, numerator, ratio;

numerator = 5;

denominator = 0;

**try** {

ratio = numerator/denominator;

System.***out***.println(ratio);

} **catch** (ArithmeticException e) {

System.***out***.println("Divide by zero");

e.printStackTrace();

}

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

//

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

// }

}

}

**Example2**

**public** **class** Example1

{

**public** **static** **void** main(String[] args)

{

**int** i, ratio;

**int**[] numbers = {100,10,0,5,2,8,0,30};

**for** (i=0; i < numbers.length-1; i++) {

**try** {

ratio = numbers[i] / numbers[i+1];

System.***out***.println(numbers[i]+"/"+numbers[i+1]+"="+ratio);

} **catch** (ArithmeticException ae)

{

System.***out***.println("Couldn't calculate "+ numbers[i]+"/"+numbers[i+1]);

}

}

// try

// {

// for (i=0; i < numbers.length-1; i++)

// {

// ratio = numbers[i] / numbers[i+1];

// System.out.println(numbers[i]+"/"+numbers[i+1]+"="+ratio);

// }

// }

// catch (ArithmeticException ae)

// {

// System.out.println("Couldn't calculate "+ numbers[i]+"/"+numbers[i+1]);

// }

}

}

**Example3**

**public** **class** Example1

{

**public** **static** **void** main(String[] args)

{

**int** i;

**int**[] data = {50, 320, 97, 12, 2000};

**for** (i=0; i < 5; i++)

{

System.***out***.println(data[i]);

}

System.***out***.println("Done");

// try

// {

// for (i=0; i < 10; i++)

// {

// System.out.println(data[i]);

// }

// }

// catch (ArrayIndexOutOfBoundsException aioobe)

// {

// System.out.println("Done");

// }

}

}

**Example4**

**import** java.util.\*;

**public** **class** Example1

{

**public** **static** **void** main(String[] args)

{

**double** leftOperand, result, rightOperand;

String leftString;

String operator;

String rightString;

StringTokenizer tokenizer;

Scanner in = **new** Scanner(System.***in***);

String str = in.nextLine();

tokenizer = **new** StringTokenizer(str, "+", **true**);

StringTokenizer token = **new** StringTokenizer(in.nextLine(), "-", **true**);

StringTokenizer tokeni = **new** StringTokenizer(in.nextLine(), "\*", **true**);

StringTokenizer tokeniz = **new** StringTokenizer(in.nextLine(), "/", **true**);

Vector<Double> v = **new** Vector<Double>();

**if** (str.contains("+")) {

leftString = tokenizer.nextToken();

operator = tokenizer.nextToken();

rightString = tokenizer.nextToken();

leftOperand = Double.*parseDouble*(leftString);

rightOperand = Double.*parseDouble*(rightString);

} **else** **if** (str.contains("-")) {

leftString = token.nextToken();

operator = token.nextToken();

rightString = token.nextToken();

leftOperand = Double.*parseDouble*(leftString);

rightOperand = Double.*parseDouble*(rightString);

} **else** **if** (str.contains("\*")) {

leftString = tokeni.nextToken();

operator = tokeni.nextToken();

rightString = tokeni.nextToken();

leftOperand = Double.*parseDouble*(leftString);

rightOperand = Double.*parseDouble*(rightString);

}

**else** {

leftString = tokeniz.nextToken();

operator = tokeniz.nextToken();

rightString = tokeniz.nextToken();

leftOperand = Double.*parseDouble*(leftString);

rightOperand = Double.*parseDouble*(rightString);

}

**try**

{

**int** n = in.nextInt();

**for** (**int** i = 0; i < n; i++) {

**if** (operator.equals("+")) {

leftString = tokenizer.nextToken();

operator = tokenizer.nextToken();

rightString = tokenizer.nextToken();

leftOperand = Double.*parseDouble*(leftString);

rightOperand = Double.*parseDouble*(rightString);

result = leftOperand + rightOperand;

v.add(result);

}

**else** **if** (operator.equals("-")) {

leftString = token.nextToken();

operator = token.nextToken();

rightString = token.nextToken();

leftOperand = Double.*parseDouble*(leftString);

rightOperand = Double.*parseDouble*(rightString);

result = leftOperand - rightOperand;

v.add(result);

}

**else** **if** (operator.equals("\*")) {

leftString = tokeni.nextToken();

operator = tokeni.nextToken();

rightString = tokeni.nextToken();

leftOperand = Double.*parseDouble*(leftString);

rightOperand = Double.*parseDouble*(rightString);

result = leftOperand \* rightOperand;

v.add(result);

}

**else** **if** (operator.equals("/")) {

leftString = tokeniz.nextToken();

operator = tokeniz.nextToken();

rightString = tokeniz.nextToken();

leftOperand = Double.*parseDouble*(leftString);

rightOperand = Double.*parseDouble*(rightString);

result = leftOperand / rightOperand;

v.add(result);

}

**else** {

result = 0.0;

}

v.add(result);

}

**for** (**int** i = 0; i < v.size(); i++) {

System.***out***.println("Result: " + v.get(i));

}

}

**catch** (NoSuchElementException nsee)

{

System.***out***.println("Invalid syntax");

}

**catch** (NumberFormatException nfe)

{

System.***out***.println("One or more operands is not a number");

}

}

}