**Student Name: Kulyash Konyrova**

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 Example1.main(Example1.java:11)

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

Because when it started running a program, it tries to divide by zero, but it can’t do this.

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?**Exception in thread "main" java.lang.Error: Unresolved compilation problem:

The local variable ratio may not have been initialized

at Example1.main(Example1.java:16)

Because it can't do the division in try block, it run the catch block, but it is empty.

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.

Done.

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.

java.lang.ArithmeticException: / by zero

Done.

at Example1.main(Example1.java:12)

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

I think yes, because it shows that there is a problem (dividing by zero).

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?**

Exception in thread "main" java.lang.Error: Unresolved compilation problems:

The local variable i may not have been initialized

The local variable i may not have been initialized

at Example2.main(Example2.java:19)

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

* Compile Example2.

**What error was generated?**

Exception in thread "main" java.lang.Error: Unresolved compilation problems:

The local variable i may not have been initialized

The local variable i may not have been initialized

at Example2.main(Example2.java:21)

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

Because

* 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?**

Because when it comes to divide by zero, program runs a catch block and don’t return to try block again.

* 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".

50

320

97

12

2000

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 changed the last index in a for loop.

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.**

It divides one long string to several strings (word by word). It breaks string “My name is Kulyash” in “My”, “name”, “is”, “Kulyash” strings.

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

A string, that would be parsed. String, which would divide tokens(for example, for word-by-word division we write “ “) and Boolean true or false;

* Run Example4.java.

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

**What output is generated?**

5.3+9.2

Result: 14.5

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

**What output is generated?**

5.3+

Invalid syntax

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

NoSuchElementException

Because there is no rightOperand.

* Run again and enter 5.3+a.

**What output is generated?**

5.3+a

One or more operands is not a number

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

NumberFormatException

Because a rightOperand isn't 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**

**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);

}

**catch** (ArithmeticException err) {

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

err.printStackTrace();

}

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

}

}

**Example2**

**public** **class** Example2

{

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

{

**int** i, ratio;

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

i = 0;

**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]);

}

}

}

}

**Example3**

**public** **class** Example3

{

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

{

**int** i;

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

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

{

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

}

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

}

}

**Example4**

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

**public** **class** Example4

{

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

{

**double** leftOperand, result, rightOperand;

String leftString, operator, rightString;

StringTokenizer tokenizer1, tokenizer2;

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

tokenizer1 = **new** StringTokenizer(in.nextLine(), " ", **false**);

tokenizer2 = **new** StringTokenizer(in.nextLine(), "+-/\*", **true**);

**while** (tokenizer1.hasMoreTokens()) {

tokenizer1.nextToken();

**try**

{

leftString = tokenizer2.nextToken();

operator = tokenizer2.nextToken();

rightString = tokenizer2.nextToken();

leftOperand = Double.*parseDouble*(leftString);

rightOperand = Double.*parseDouble*(rightString);

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

result = leftOperand + rightOperand;

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

result = leftOperand - rightOperand;

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

result = leftOperand \* rightOperand;

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

result = leftOperand / rightOperand;

**else**

result = 0.0;

System.***out***.println("Result: " + result);

}

**catch** (NoSuchElementException nsee)

{

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

}

**catch** (NumberFormatException nfe)

{

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

}

}

}

}