



Exceptions and File Handling

Objective:

- Practice on avoiding exception
- Practice on catching exceptions
- Practice on defining a custom exception class
- Practice on handling files (reading and writing)

Introduction

Exceptions in Java is one of the important concepts that you have to learn how to deal with it whether you like it or not. Because every program has to take care of its errors, it is a must that you learn the type of errors in the language that you are coding with. Errors In Java are either exceptions or errors. Although both of them can crash your program, but your program can avoid at least exceptions or catch it.

Part#1 avoiding exceptions

In Java you can either wait for the exception to happen or you can simply avoid the exception occurrences. The simplest way to avoid exceptions is to write a code that checks abnormal data. Using the if statement, you can write conditions that check whether the data is what you want it to be or not. Therefore, you have to think about all the possible data that every variable could hold.

In the keyboard simulator program, there are a lot of variables that can hold wrong data. Most of those data are user input. The first one, is when the user enters a keyboard code. To avoid having exceptions in your program coming from this input, you have to check if the code that the user entered is available. The simplest way to check wrong inputs is by checking if the entered code is one of keyboard codes you provided for the user. The other way, is by letting the program look for the code. If the code is not found, then the program will return null. Both of these checking methods works perfectly, but some of them are more general than the other. If you check if the input is one of the keyboard codes, then you are not checking if the input is not a number. Therefore, checking if the program returns a null pointer after button look up is better for this case, since this method is going to cover abnormal inputs like letters and symbols. Your program should deal with the problem like in the picture below.

LAB4 1 Exception

1-QWERTY

2-Claculator

Please choose one of the two keyboards to run:

1

| | | | |
|-------|-------|-------|---------------|
| 1->a | 11->k | 21->u | 31->4 |
| 2->b | 12->l | 22->v | 32->5 |
| 3->c | 13->m | 23->w | 33->6 |
| 4->d | 14->n | 24->x | 34->7 |
| 5->e | 15->o | 25->y | 35->8 |
| 6->f | 16->p | 26->z | 36->9 |
| 7->g | 17->q | 27->0 | 37->Space |
| 8->h | 18->r | 28->1 | 38->New Line |
| 9->i | 19->s | 29->2 | 39->Backspace |
| 10->j | 20->t | 30->3 | |

Please enter the code of the button or -1 to exit: 0

Not Found!

Please enter the code of the button or -1 to exit: 40

Not Found!

Please enter the code of the button or -1 to exit: s

Not Found!

Please enter the code of the button or -1 to exit: *

Not Found!

Please enter the code of the button or -1 to exit: "khalid"

Not Found!

Please enter the code of the button or -1 to exit:

The next input that you need to check is the keyboard choosing option. The user is going to choose whether he want to use the QWERTY or the calculator. You are simply asking the user to enter 1 or 2. If you are using the nextInt() method to get the user input, then we come to the same problem we discussed before. The program is not able to deal with the unexpected import like letters and symbols. Therefore, an exception will be thrown. To avoid this, you can use the next() method which accept numbers, letters, and symbols without any exception thrown. Then check if the user enters 1 or 2.

LAB4 2 Exception

1-QWERTY

2-Claculator

Please choose one of the two keyboards to run:

0

1-QWERTY

2-Claculator

Please choose one of the two keyboards to run:

QWERTY

1-QWERTY

2-Claculator

Please choose one of the two keyboards to run:

1

| | | | | |
|-------|-------|-------|---------------|----------|
| 1->a | 11->k | 21->u | 31->4 | 41->Save |
| 2->b | 12->l | 22->v | 32->5 | |
| 3->c | 13->m | 23->w | 33->6 | |
| 4->d | 14->n | 24->x | 34->7 | |
| 5->e | 15->o | 25->y | 35->8 | |
| 6->f | 16->p | 26->z | 36->9 | |
| 7->g | 17->q | 27->0 | 37->Space | |
| 8->h | 18->r | 28->1 | 38->New Line | |
| 9->i | 19->s | 29->2 | 39->Backspace | |
| 10->j | 20->t | 30->3 | 40->Load | |


Please enter the code of the button or -1 to exit:

Finally, another more exception to avoid is where the user is trying to use the backspace when there is no text. If the use does that, a `StringIndexOutOfBoundsException` is thrown. The simple fix can be achieved using the if statement to check if the displayed text is empty or not.

```
1-QWERTY
2-Calculator
Please choose one of the two keyboards to run:
1
1->a | 11->k | 21->u | 31->4 | 41->Save |
2->b | 12->l | 22->v | 32->5 |
3->c | 13->m | 23->w | 33->6 |
4->d | 14->n | 24->x | 34->7 |
5->e | 15->o | 25->y | 35->8 |
6->f | 16->p | 26->z | 36->9 |
7->g | 17->q | 27->0 | 37->Space |
8->h | 18->r | 28->1 | 38->New Line |
9->i | 19->s | 29->2 | 39->Backspace |
10->j | 20->t | 30->3 | 40->Load |
Please enter the code of the button or -1 to exit: 39
Texted entered:
-----
-----
Please enter the code of the button or -1 to exit: 39
Texted entered:
-----
-----
Please enter the code of the button or -1 to exit:
```

Part2 caching exceptions

There are some situations where your program is executing mathematical expression like the code in the calculator class. Unlike the situations we covered before, errors coming from mathematical expressions can be trickier to avoid. Therefore, you have to deal with exceptions. As we discussed in class, exceptions are thrown, so you have to catch them. In the calculator class, you are calling the `evaluate` method in the `EvaluateString` class and it might throw exceptions while trying to evaluate the expression Entered by the user. The `UnsupportedOperationException` is throwing when the user is trying to divide by zero. Since you're not allowed to modify the `EvaluateString` class, you have to use the `try` statement to call the evaluation method and the `catch` statement to catch the `UnsupportedOperationException`. The following picture shows the problem.




```

Please choose one of the two keyboards to run:
2
0->0 10->+
1->1 11->-
2->2 12->/
3->3 13->*
4->4 14->=
5->5 15->(
6->6 16->)
7->7 17->Space
8->8 18->New Line
9->9 19->Backspace
Please enter the code of the button or -1 to exit: 1
Texted entered:
-----
1
-----
Please enter the code of the button or -1 to exit: 12
Texted entered:
-----
1/
-----
Please enter the code of the button or -1 to exit: 0
Texted entered:
-----
1/0
-----
Please enter the code of the button or -1 to exit: 14
Exception in thread "main" java.lang.UnsupportedOperationException: Cannot divide by zero
    at LAB4_1_Exception.EvaluateString.applyOp(EvaluateString.java:143)
    at LAB4_1_Exception.EvaluateString.evaluate(EvaluateString.java:103)
    at LAB4_1_Exception.Calculator.buttonPressed(Calculator.java:41)
    at LAB4_1_Exception_OS.OperatingSystem.main(OperatingSystem.java:47)

```

The handle for the UnsupportedOperationException Should tell the user that this is unacceptable expression and the program is going to delete it. Then the program is going to delete ONLY the last expression.




```

Texted entered:
-----
1+1= 2
-----
Please enter the code of the button or -1 to exit: 1
Texted entered:
-----
1+1= 2
1
-----
Please enter the code of the button or -1 to exit: 12
Texted entered:
-----
1+1= 2
1/
-----
Please enter the code of the button or -1 to exit: 0
Texted entered:
-----
1+1= 2
1/0
-----
Please enter the code of the button or -1 to exit: 14
***** Unacceptable expression! I will delete it *****
Texted entered:
-----
1+1= 2
-----
Please enter the code of the button or -1 to exit:


```

There is another exception that can be thrown when trying to evaluate the mathematical expression and it is called EmptyStackException. It happens when there aren't enough operands for

every operator in the expression or there are no operations in the expression. The following two picture shows both scenarios.



```
LAB4_1_Exception
1-QWERTY
2-Claculator
Please choose one of the two keyboards to run:
2
0->0 | 10->+      |
1->1 | 11->-      |
2->2 | 12->/      |
3->3 | 13->*      |
4->4 | 14->=      |
5->5 | 15->(      |
6->6 | 16->)      |
7->7 | 17->Space  |
8->8 | 18->New Line|
9->9 | 19->Backspace|
Please enter the code of the button or -1 to exit: 14
Exception in thread "main" java.util.EmptyStackException
    at java.util.Stack.peek(Unknown Source)
    at java.util.Stack.pop(Unknown Source)
    at LAB4_1_Exception.EvaluateString.evaluate(EvaluateString.java:109)
    at LAB4_1_Exception.Calculator.buttonPressed(Calculator.java:41)
    at LAB4_1_Exception_OS.OperatingSystem.main(OperatingSystem.java:47)
```

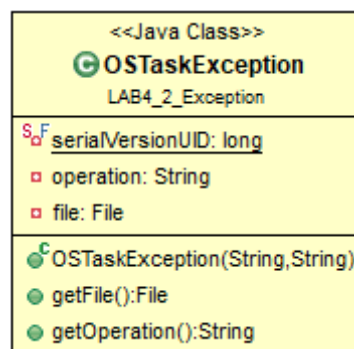


```
LAB4_1_Exception
1-QWERTY
2-Claculator
Please choose one of the two keyboards to run:
2
0->0 | 10->+      |
1->1 | 11->-      |
2->2 | 12->/      |
3->3 | 13->*      |
4->4 | 14->=      |
5->5 | 15->(      |
6->6 | 16->)      |
7->7 | 17->Space  |
8->8 | 18->New Line|
9->9 | 19->Backspace|
Please enter the code of the button or -1 to exit: 1
Texted entered:
-----
1
-----
Please enter the code of the button or -1 to exit: 10
Texted entered:
-----
1+
-----
Please enter the code of the button or -1 to exit: 14
Exception in thread "main" java.util.EmptyStackException
    at java.util.Stack.peek(Unknown Source)
    at java.util.Stack.pop(Unknown Source)
    at LAB4_1_Exception.EvaluateString.evaluate(EvaluateString.java:105)
    at LAB4_1_Exception.Calculator.buttonPressed(Calculator.java:41)
    at LAB4_1_Exception_OS.OperatingSystem.main(OperatingSystem.java:47)
```

Part3 defining new exception

Exceptions are not always errors. They can be used to switch control during the runtime of the program. In other words, methods that throws exceptions are telling the caller that it is not going to deal with current situation. The caller of the method has to figure out the situation that the called method does not want to deal with. The only way to understand the situation is by looking into the thrown exception. Programmers usually check the name of the thrown exception to understand the undealt with situation. Furthermore, programmers use the methods available by the thrown exception object to get more information. For example, `printStackTrace()` prints the line number where the exception happened, the call stack and the exception name. However, there might be more information you want to tell the caller of the method.

In this part of the lab, you will create a class that extends the exception class called `OSTaskException`. This exception is going to be thrown whenever the keyboard is switching control to the operating system. The situation that keyboard class is not going to deal with is the file handling. Therefore, the keyboard has to provide some information for the operating system to handle the file loading and saving. After the user choose the operation, Loading or saving, from the keyboard menu, the keyboard should create a new `OSTaskException` object. The `OSTaskException` object holds the file path of the file, entered by the user and the operation type as shown in the following UML.



Since the Operating system class calls the `buttonPressed()` method in the keyboard class and the `buttonPressed` method calls the `act` method in the `SpecialAcrionButton` class, the `act` method should be the one who throws the `OSTaskException`. The `act` method accepts a copy of the displayed text and from the type of the special button it will determine the requested operation. The operations handled by the `act` method are Backspace, Load, and Save (notice: the Operating system is the one who adds them to the keyboard and the keyboard is the one who is performing them).

If the requested operation is load, then the keyboard asks the user to enter the file path. Then it will create the `OSTaskException` and throw it. The `buttonPressed()` method in the keyboard class will also throws the same exception and the operating system class should handle this exception. Now, the control is handled by the operating system to open the file and read. Since the requested operation is loading, then the operating system will start by reading the file using the scanner object. The operating system will also use the `try with a resource` to try to open the file and catch the `FileNotFoundException` If occurred. After reading the content of the file, the operating system will call the new `addLoadedContent()` method in the Keyboard class to add the new text. This method will display to the user the loaded content of the file and ask if the user wants to append, replace, or ignore the loaded content of the file. Then it updates the content of the displayed text in the keyboard.

```

Please enter the code of the button or -1 to exit: 40
Keyboard: Please enter the file path :
TestFile
OS: I will handle this
***** Loaded *****
test
***** DONE *****
Keyboard: what do you want to do with the new Text :
1-append.
2-replace.
3-ignore
1
Done adding.
Texted entered:
-----
test
-----
Please enter the code of the button or -1 to exit:

```

If the requested operation is save, then the keyboard asks the user to enter the file path. Then the control is going to switch to the operating system to open the file and write on it. Then the operating system catches the exception and using the operation name stored in the thrown exception object it will distinguish if the requested operation is save or load. After the operating system finds out that the requested operation is saving, it will prepare to write by creating an object from the FileWriter class that will write the text stored in the Keyboard class. Finally, The OS writes the new content of the file required using try and catch block to catch the IOException.

```

Texted entered:
-----
test
-----
Please enter the code of the button or -1 to exit: 41
Keyboard: Please enter the file path :
TestFile
OS: I will handle this
writing...
***** Saved *****
test
***** DONE *****
Please enter the code of the button or -1 to exit:

```

What to turn in

You should turn your work in a **MS Word** file that has your codes for each part as specified below. All the codes should be yours. **NO COPY/PASTE IS ALLOWED**. You should always copy/paste the code from your compiler and take a screenshot of the console.

Part#1:

- The code of all the exception avoiding solutions.
- The screenshot of the output shown a successful exception avoiding.

Part#2:

- The for all the try-catch statements and the handling code.
- The screenshot of the output shown a successful exception catching and handling.

Part#3:

- The new OSTaskException class
- All the classes where you made modifecations to use the OSTaskException including try-catch in OS class.
- The screenshot of the output shown a successful exception catching and handling.
- The final version of you project in one zip file that runs without any errors.