**# Programming languages / Java (BSc, 18) 5th lab**

### Task 1

Modify the solution of \*\*Task 7\*\* of the \*\*1st lab\*\* so that operands have type `double` and the operations to be performed are provided as command line arguments. No need to extend the set of supported operations.

In case the program is called with insufficient number of arguments, it shall throw an `IllegalArgumentException` in its `main()` method. Throw an `ArithmeticException` in case of diving by zero and, in case of an unsupported arithmetic operation also throw an `IllegalArgumentException`.

Some related links:

<https://www.tutorialspoint.com/java/switch_statement_in_java.htm>

<https://www.tutorialgateway.org/java-charat-method/>

https://www.javatpoint.com/exception-handling-in-java

class Calculator {

public static void main(String[] args){

if (args.length != 3){

throw new IllegalArgumentException();

} else {

double a = Double.parseDouble(args[0]);

char m = args[1].charAt(0); //why it is charAt(0)?

double b = Double.parseDouble(args[2]);

System.out.print("" + a + m + b + " = ");

switch (m) {

case '+':

System.out.println(a + b);

break;

case '-':

System.out.println(a - b);

break;

case '\*':

System.out.println(a \* b);

break;

case '/':

if (b == 0) {

throw new ArithmeticException();

}

System.out.println(a / b);

break;

default:

throw new IllegalArgumentException();

}

}

}

}

### Task 2

Modify \*\*Task 1\*\* so that exceptions are caught in `main()` and appropriate error messages are printed to the console instead (eg:`"Invalid program`

`arguments provided."`, in case of a missing argument). Also catch the exception thrown by `parseDouble()` in case of an invalid string.

<https://www.geeksforgeeks.org/double-parsedouble-method-in-java-with-examples/>

<https://www.javatpoint.com/try-catch-block>

class Calculator {

public static void main(String[] args){

try {

if (args.length != 3){

throw new IllegalArgumentException();

} else {

double a = Double.parseDouble(args[0]);

char m = args[1].charAt(0);

double b = Double.parseDouble(args[2]);

System.out.print("" + a + m + b + " = ");

switch (m) {

case '+':

System.out.println(a + b);

break;

case '-':

System.out.println(a - b);

break;

case '\*':

System.out.println(a \* b);

break;

case '/':

if (b == 0) {

throw new ArithmeticException();

}

System.out.println(a / b);

break;

default:

throw new IllegalArgumentException();

}

}

}

catch (NumberFormatException e) {

System.out.println("Invalid number format.");

}

catch (IllegalArgumentException e) {

System.out.println("Invalid program arguments provided.");

}

catch (ArithmeticException e) {

System.out.println("Arithmetic error occurred.");

}

}

}

### Task 3

Modify \*\*Task 2\*\* so that each exception thrown by you is provided with an appropriate error message in its constructor. When catching these exceptions print \*\*these\*\* messages to the console.

https://www.javatpoint.com/post/java-throwable-getmessage-method

class Calculator {

public static void main(String[] args){

try {

if (args.length != 3){

throw new IllegalArgumentException("Too few arguments provided.");

} else {

double a = Double.parseDouble(args[0]);

char m = args[1].charAt(0);

double b = Double.parseDouble(args[2]);

System.out.print("" + a + m + b + " = ");

switch (m) {

case '+':

System.out.println(a + b);

break;

case '-':

System.out.println(a - b);

break;

case '\*':

System.out.println(a \* b);

break;

case '/':

if (b == 0) {

throw new ArithmeticException("Division by zero.");

}

System.out.println(a / b);

break;

default:

throw new IllegalArgumentException("Unknown operation.");

}

}

}

catch (NumberFormatException e) {

System.out.println("Invalid number format.");

System.out.println(e.getMessage());

}

catch (IllegalArgumentException e) {

System.out.println("Invalid program arguments provided.");

System.out.println(e.getMessage());

}

catch (ArithmeticException e) {

System.out.println("Arithmetic error occured.");

System.out.println(e.getMessage());

}

}

}

### Task 4

We have an input file containing integer numbers separated by commas. Your task is to calculate the \*\*sum\*\* of values in each line and print result (lines of sums) to another file.

```

1,2,5,-2

10,20,0,7

3,2

2

0

1,2

3

```

Use `BufferedReader` in your solution!

Look for a method in the `String` class that is capable of splitting a line around some special character (comma, in our case)!

We can assume the input file's format is always valid.

Some links:

<https://www.javatpoint.com/java-bufferedreader-class>

<https://www.javatpoint.com/java-file-class>

https://www.javatpoint.com/java-filereader-class

import java.io.\*;

public class Main {

public static void main(String[] args) {

if (args.length < 2) {

System.err.println("Not enough command line arguments. 2 required.");

}

String inputFile = args[0];

String outputFile = args[1];

summarize(inputFile, outputFile);

}

private static void summarize(String inputFile, String outputFile) {

File in = new File(inputFile);

File out = new File(outputFile);

try (BufferedReader br = new BufferedReader(new FileReader(in));

PrintWriter pw = new PrintWriter(out) ) {

String line;

while ( null != (line = br.readLine()) ) {

String[] parts = line.split(",");

int sum = 0;

for (String number : parts) {

int i = Integer.parseInt(number);

sum += i;

}

pw.println(sum);

}

} catch (IOException e) {

System.err.println("An IO error occurred.");

}

}

}

### Task 5

Look up some string in a text file! The file name is provided as command line argument, the string should be read from the console.

Print the number of lines containing an occurrence of the string to the console!

Partial matches also count. That is, the line can contain other text beside our string and still count as an occurrence. (Try finding a relevant method in the `String` class!)

Some links:

<https://www.javatpoint.com/post/java-scanner-hasnextline-method>

https://www.w3schools.com/JAVA/ref\_string\_contains.asp

```

import java.io.\*;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

if (args.length < 1) {

System.err.println("Not enough command line arguments. 1 required.");

}

String inputFile = args[0];

System.out.print("Text to find: ");

String textToFind = System.console().readLine();

countAppearances(inputFile, textToFind);

}

private static void countAppearances(String inputFile, String textToFind) {

File in = new File(inputFile);

try ( Scanner sc = new Scanner(in) ) {

int count = 0;

while (sc.hasNextLine() ) {

String line = sc.nextLine();

if (line.contains(textToFind)) {

++count;

}

}

System.out.println("'" + textToFind + "' appeared in " +

count + " lines.");

} catch (FileNotFoundException e) {

System.err.println("The file cannot be opened.");

}

}

}

Link Related to Lab4:

https://www.javatpoint.com/StringBuilder-class

https://www.techiedelight.com/difference-between-string-stringbuilder-java/