CIT 130 Beginning Java Spring 2022

Assignment #12 – Total Points = 100

Assigned Date: 4/18 Due date: 4/26

DO NOT share your answers with anyone. DO NOT collaborate on completing work with anyone. DO NOT use the Internet to search for specific solutions to assignments. DO NOT pay anyone to complete your assignment. **Avoid web sites that offer solutions to assignments. If you copy work from such web sites, keep in mind that other students are also looking at the same information and will therefore submit duplicated work.** Failure to meet these requirements leads to a violation of the academic integrity principles as stated in your syllabus.

Grading Criteria: See the grading criteria information in your course **syllabus addendum (**Click Home and look towards the middle of the page**)**.

Objective: Demonstrate your understanding of using exceptions and files in Java**.** This assignment is based on the material covered in **chapter 12** of your textbook.

Deliverable: Use the file format:

- **firstNameLastNamecit130_Pa12.java** for the source file of your class. You should only submit one Java source file. There is no need to create a utility class for this exercise.

Documentation: See your course **syllabus addendum** file in Canvas for the required heading documentation and other notes. Make sure you have the header section properly created using tags such as @author, @file, etc. **Methods need to be properly documented**. See the syllabus addendum file, part 2 (e) on proper process to document each method.

Assignment: Write a program to practice with the use of exceptions and text files in Java.

Constraints:

- Do not use break/continue in your loops.
- Do not use the System.exit() method to exit the program.
- Do not use global variables. The only global variable allowed would be the definition of the Scanner class, or in very specific situations as strictly required by the program requirements.

Process:

0. Write a **method** that returns a valid integer. The function does not have a parameter. In our case, a valid integer is any whole number. Use the try/catch statements to handle the error in data entry. Using a loop, make sure to ask the user for a valid integer.
1. Write a **method** to provide a menu of options. Make sure to call the method in step 0 (above) to get an integer for the menu choice.

2. Write a **method** that returns a valid decimal number. In our case a valid decimal number is any number between 1 and 100. For example, 3.8, 90, and 2.789 are all valid input. Use the try/catch statements to handle the error in data entry. Using a loop, make sure to ask the user for a valid number.
3. Write a **method** to perform the following tasks. The method does not return a value and does not have any parameters.
    - Create an array with 10 random **integers** between 10 and 100, inclusive.
    - Display the content of the array.
    - Prompt the user to enter an index number of the array and display the corresponding element value. If the specified index is out of bounds, handle the exception and display the message **out of bounds.** HINT: **IndexOutOfBoundsException** will catch such errors. For example, if the user enters -1 or 12 for array index value, an exception must be thrown.
    - Ask the user for the name of an output file. Write your **own full name** (do not ask for input) as the first line of the file and write the contents of this array to this file.
4. Write a **method** to perform the following tasks:
    - I have provided a text file named **testData.txt.** It contains a string of characters. You can also test your code with the attached file of the book "pride and prejudice" (**pride_and_prejudice.txt**). Write code to verify if the file exists using **try/catch** block. If the file exists, calculate, and display the number of **vowels** (a, e, i, o, u – case insensitive), the number of **digits** in the file., and the size of the file in bytes. Hard-code the name of the file as **textData.txt**. If the file does not exist, display an error message. "hard-code" simply means to assign the name of the file as the constant "testData.txt".
    - NOTE: Simply download the file into your local folder where your source file exists in order to use it for your testing.
5. Write a main method that tests the functionality of these methods. Make sure to display the data to properly test your code. See the sample interaction for an example.

Sample Interaction

Enter 1 to validate an integer

Enter 2 to validate a decimal number

Enter 3 to process an array

Enter 4 to process a file

Enter 5 to exit

abc

*** Invalid entry - Try again

2.4

*** Invalid entry - Try again

1

Enter an integer to test, more than -5: -9

*** Invalid entry - Try again

a

*** Invalid entry - Try again

2.8

*** Invalid entry - Try again

42

You entered a valid integer: 42

Enter 1 to validate an integer

Enter 2 to validate a decimal number

Enter 3 to process an array

Enter 4 to process a file

Enter 5 to exit

2

Enter a decimal number: a1

*** Invalid entry - Try again

123

*** Invalid entry - Try again

-1

*** Invalid entry - Try again

4.9

You entered a valid decimal number: 4.9

Enter 1 to validate an integer

Enter 2 to validate a decimal number

Enter 3 to process an array

Enter 4 to process a file

Enter 5 to exit

3

Array data:

74 17 82 25 22 72 44 41 29 33

Enter a index of the array: 12

Error, invalid index was entered.

Enter a filename to save array: 1.txt

Enter 1 to validate an integer

Enter 2 to validate a decimal number

Enter 3 to process an array

Enter 4 to process a file

Enter 5 to exit

3

Array data:

15 52 41 23 81 5 88 31 20 33

Enter a index of the array: a1

*** Invalid entry - Try again

0

The index 0 Has 15 stored.

Enter a filename to save array: 2.txt

Enter 1 to validate an integer

Enter 2 to validate a decimal number

Enter 3 to process an array

Enter 4 to process a file

Enter 5 to exit

3

Array data:

39 9 48 21 56 76 40 21 35 7

Enter a index of the array: -1

Error, invalid index was entered.

Enter a filename to save array: 3.txt

Enter 1 to validate an integer

Enter 2 to validate a decimal number

Enter 3 to process an array

Enter 4 to process a file

Enter 5 to exit

4

Number of vowels: 33

Number of Digits: 8

The file has 124 bytes

Enter 1 to validate an integer

Enter 2 to validate a decimal number

Enter 3 to process an array

Enter 4 to process a file

Enter 5 to exit

5

NOTE THAT you should have 3 files named 1.txt, 2.txt, and 3.txt with data in it once we are done running this code.