# CST8110 Assignment #4 [5%]

**This assignment relates to the following Course Learning Requirements:**

CLR 1: Define, analyze, and document the logic of a solution to a given problem.

CLR 2: Implement the solution to a given problem by writing the appropriate code in a high-level language (Java).

CLR 3: Explore the difference between arrays of primitives and arrays of references.

CLR 4: Install and use the Java Development, Runtime Environment, and documentation libraries.

CLR 7: Create documentation and a Java solution for programming problems that adhere to the submission standard identified within the timeframe given in the problem description.

Objective of this Assignment:

The following is an exercise to help you understand what you have learned in this module. It will allow you to familiarize yourself with the module content as well as practice the skills required to develop software using the Java programming language. The objective of this assignment is to write a class named StringToIntArray that will convert a String containing multiple numeric values into an array of integers. To do this, you will use the Scanner class in a new way – instead of reading from the standard input, you will read it from a String.  You will be calling methods within the class itself to help reduce repeated code. Re-using methods can reduce development time and make code easier to maintain.

There are several ways to design the logic for this assignment, however if the logic gets the correct result, it is correct!

# Pre-Assignment Instructions:

1. To prepare you for this assignment, please complete the content associated to this module.
2. Complete all module reading and learning activities
3. Complete all reference video tutorials
4. Complete Lab 5
5. Develop this program locally on your machine in a plain text editor or other IDE.

**Assignment Tasks/Steps:**

As a programmer, it is very common to receive data in one form and need to convert it to another form before it is used.

1. The **Arrays** class may be referenced in your code is in the **import** statement and when using the **fill** method to populate the entire array with **Integer.MIN\_VALUE**.
2. Create a Java class named **StringToIntArray**.
   1. Properties:
      1. **intArray:** a private array of 10 integer values, **index**: int value for the index of array.
   2. Methods**:**
      1. **public StringToIntArray():**a no-args constructor that uses the static fill method of the **java.util.Arrays** class to populate every element of intArray with the constant value Integer.MIN\_VALUE. If an element contains Integer.MIN\_VALUE, you may consider it to be “empty”.
      2. Add the main method to ensure program compiles and runs. The user should be prompted to input a string(containing int’s in between the string) and the program needs to print the output array(containing only int’s) on screen. The logic for input and output can be implemented in main method.
      3. Define and initialize Boolean variable **skipErrors(true or false).** Define and initialize String variable **str.** Create an **object** of class StringToIntArray. Get the input string from user using **Scanner** class and assign it to string variable str. Pass the input received from user and skipErrors as parameters to **scanStringToIntArray**  as a method call via the object of class StringToIntArray.
      4. **public boolean scanStringToIntArray(String s, boolean skipErrors):**
         1. Declare a local **Scanner** variable, using the String **s** as a parameter to the Scanner constructor (instead of System.in)
         2. **While** the **hasNext()** method of the Scanner returns true, **if** the next value in the Scanner contains an integer put it in the intArray
            1. Use the **hasNextInt()** method of the Scanner to validate the presence of an integer, and the nextInt()method of the **Scanner** to read in the value
         3. If value of skipErrors is chosen as false, if any values are encountered in the String s which are not integers, intArray should remain (or be returned to) an array of 10 elements of Integer.MIN\_VALUE and false should be returned
         4. If value of skipErrors is chosen as true, if any values encountered in the String s which are not integers, the non-integer value should simply be skipped over and intArray should continue populating with the next String value
         5. **Hint:** If a value is not an integer, don’t forget to read and discard the value using the next() method of the Scanner
   3. Test case when skiperrors is “true”(preferred):
      1. Please enter a string:

s 1 2 3 o 3 4 6

[1, 2, 3, 3, 4, 6, -2147483648, -2147483648, -2147483648, -2147483648]”

* + 1. Please enter a string:

s123o 3 4 6

[3, 4, 6, -2147483648, -2147483648, -2147483648, -2147483648, -2147483648, -2147483648, -2147483648]

Test case when skiperrors is “false:

s123o 3 4 6

[-2147483648, -2147483648, -2147483648, -2147483648, -2147483648, -2147483648, -2147483648, -2147483648, -2147483648, -2147483648]

* 1. As an extended test case, if you feel like exploring some more, you can try to implement the following test case when skiperrors is “true”:  
     You're not expected to fulfill this test case and won’t get extra credit but if you like to explore and implement a challenging code on your own, you can go for it...
     1. Input string: “s123o 3 4 6”

[1, 2, 3, 3, 4, 6, -2147483648, -2147483648, -2147483648, -2147483648]

**Submit Your Assignment**

Your assignment is to be submitted on Brightspace as Java files. It should be submitted with the following guidelines:

1. Include the file header using the template provided in Assignment 1, in every java file submitted.
2. Follow expected style guidelines:
   * 1. Use “Egyptian” style braces for all classes and methods
     2. Indent your code using 4 spaces (no tabs)
     3. Follow naming conventions for all class, variable, and method identifiers
     4. Use appropriate whitespace for readability
     5. Comment your code (no less than one comment per class, and one per method).
3. File must be named **StringToIntArray.java**
4. Method names, return types, and parameters must match those in the requirements above.

Feel free to make any assumption in order to implement a working code. Make sure the code delivers expected successful output. Highlight any assumption you may make while submitting the assessment.

**Assignment Grading Rubric (5%)**

|  |  |
| --- | --- |
| Criteria / Marks | Points |
| All programs compile | /2 |
| All programs execute and output as expected from the updates performed by the learner  (i.e.: refer to section #1 and #2 to verify output from the requirements) | /20 |
| Header file is inserted at top of file and all variables and place holder and information replaced with learner’s information | /2 |
| The style guidelines are followed:  i. Use “Egyptian” style braces for all classes and methods  ii. Indent your code using 4 spaces (no tabs)  iii. Follow naming conventions for all class, variable, and method identifiers  iv. Use appropriate whitespace for readability  v. Comment your code (no less than one comment per class, and one comment per method) | /10 |
| Method names match those in the requirements exactly | /4 |
| Assignment uploaded / pushed | /2 |
| Comments |  |
| Total Points | /40 |