Session 5

**1- Revision:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Math.ceil (4.0) |  |  | | Math.floor(4.0) | |  |  | Math.round(4.0) | |  |
| Math.ceil(4.4) |  |  | | **Math.floor(4.4)** | |  |  | **Math.round(4.4)** | |  |
| Math.ceil(4.5) |  |  | | **Math.floor(4.5)** | |  |  | **Math.round(4.5)** | |  |
| Math.ceil(4.7) |  |  | | **Math.floor(4.7)** | |  |  | **Math.round(4.7)** | |  |
|  |  |  | |  | |  |  |  | |  |
| Math.max(2,3) |  |  | | **Math.sin(0)** | |  |  | **Math.abs(2)** | |  |
| Math.min(4,5.0) |  |  | | **Math.cos(0)** | |  |  | **Math.abs(-2)** | |  |
| Math.pow(2,5) |  |  | | **Math.tan(0)** | |  |  | **Math.abs(-2.6)** | |  |
| Math.sqrt(4) |  |  | |  | |  |  |  | |  |
|  |  |  | |  | |  |  |  | |  |
| Math.ceil ( 1 / 2) | | |  | | **Math.floor( Math.ceil ( 2.1 ) )** | | | |  | |
| Math.ceil ( 1.0 / 2 ) | | |  | | **Math.sqrt( Math.pow ( 2,4 ) )** | | | |  | |
| Math.ceil (1.0 / 2.0 ) | | |  | | **Math.ceil ( 1 / 2.0 )** | | | |  | |
|  | | |  | |  | | | |  | |
|  | | | | | | | | |  | |
| Math.pow ( Math.ceil ( 2.3 ) , Math.floor ( 2.9 ) ) | | | | | | | | |  | |
| Math.pow ( Math.ceil (4+3/4)-1 , Math.max( Math.max(2,3) ,5-Math.floor(3/2) ) ) | | | | | | | | |  | |

**2- Problems Test:**

It's based on a testing method called 'Black Box Testing'.

In this method we test your program as following:

**Black Box**

**Program**

**Input**

**Output**

1- Apply some input test cases.

2- Receive the program output.

3- Compare the program output to the actual output.

***This means you do NOT know everything about the input, so be prepared for all cases!***

**Problem #1:**

You will receive an array of integers; find the average of all integers in this array!

NOTES:

1-Any Java program SHOULD contain at least one class.

2- Solution will be on one method, although we can make it call other methods!

3- We may create a test method (main method) so we can test our program before submitting it!

//Solution Class

……………………….

|  |  |  |
| --- | --- | --- |
| //Solution method   |  | | --- | |  |   //Test method   |  | | --- | |  | |

**3- String**

String is any sequence of characters like "abd", "aw4& t" , "5" , "..hello world".

**A-Declaring:**

|  |
| --- |
| 1-General way  Type variable\_name = new Type (value)  Integer x = new Integer (5);  String s = new String ("Ahmed");  2-Equivalent Simple way  int x =5;  String s = "Ahmed"; |

**B- String vs. char array**

|  |  |  |
| --- | --- | --- |
|  | String | Array of characters |
| Declaration | String a = new String ("Ahmed"); | char a []= {'A', 'h' , 'm', 'e', 'd'}; |
| Size | **Dynamic size (changeable)**  That's why we get String length using a method  **a.length()** | **Fixed size (Fixed)**  That's why we get its length using **a.length;** |
| Print | System.out.print(a); | for(int i=0; i<a.length;i++)  System.out.print(a[i]) |

**C- String Concatenation:**

This means adding many strings to each other

|  |
| --- |
| String first = "Ahmed";  String last = "Hassan";  String full\_name = first+" "+last;  System.out.println (full\_name); //…………………………………………………  first+=first;  System.out.println (first); // ………………………………………… |

**D-String length IS dynamic!**

|  |  |
| --- | --- |
| String x = "Hello";  for(int i=0 ; i < x.length(); i++){  System.out.println(i+" : "+x.length());  if(i==0)  x+="World";  } | Output: |

**E-String methods //replace(char,char) toCharArray()**

|  |  |  |
| --- | --- | --- |
| SN | Methods with Description | String x="Hello";  String y = "World"; |
| 1 | [**char charAt(int index)**](http://www.tutorialspoint.com/java/java_string_charat.htm)  Returns the character at the specified index. | **x.charAt(1);**  **char c = x.charAt(0);**  **System.out.println( c );**  **System.out.println( x.charAt(5));**  **for(int i=0 ; i < x.length(); i++){**  **System.out.print(x.charAt(i));**  **}** |
| 2 | [**int compareTo(String anotherString)**](http://www.tutorialspoint.com/java/java_string_compareto.htm)  Compares two strings lexicographically. | **System.out.println(x.compareTo(y)); System.out.println("abc".compareTo("ABC"));**  **System.out.println(x.compareTo("Hello"));** |
| 3 | [**boolean equals(Object anObject)**](http://www.tutorialspoint.com/java/java_string_equals.htm)  Compares this string to the specified object. | **boolean flag = x.equals(y);**  **System.out.println(flag);**  **System.out.println(y.equals("world"));** |
| 4 | [**int indexOf(int ch)**](http://www.tutorialspoint.com/java/java_string_indexof.htm)  Returns the index within this string of the first occurrence of the specified character. | **int index = x.indexOf('H');**  **System.out.println(index);**  **System.out.println(x.indexOf('l'));**  **System.out.println(x.indexOf('r'));** |
| 5 | [**int length()**](http://www.tutorialspoint.com/java/java_string_length.htm)  Returns the length of this string. | **System.out.println(x.length());**  **String test , test2="";**  **System.out.print(test.length());**  **System.out.println(test2.length());** |
| 6 | [**String replace(char oldChar, char newChar)**](http://www.tutorialspoint.com/java/java_string_replace.htm)  Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar. | **x.replace('l', 'm');**  **System.out.println(x);**  **String z = x.replace('H', 'M');**  **System.out.println(z);**  **System.out.println(x.replace('l', 'm'));** |
| 7 | [**String substring(int beginIndex)**](http://www.tutorialspoint.com/java/java_string_substring.htm)  Returns a new string that is a substring of this string. | **System.out.println(x.substring(0));**  **System.out.println(y.substring(2));** |
| 8 | [**String substring(int beginIndex, int endIndex)**](http://www.tutorialspoint.com/java/java_string_substring.htm)  Returns a new string that is a substring of this string.  endIndex exclusive | **int a = y.length();**  **System.out.println(x.substring(0, 2));**  **System.out.println(y.substring(1, a ));** |
| 9 | [**char[] toCharArray()**](http://www.tutorialspoint.com/java/java_string_tochararray.htm)  Converts this string to a new character array. | **char ar[]=x.toCharArray();**  **for(int i=0 ; i < ar.length; i++){**  **System.out.println(ar[i]);**  **}** |
| 10 | [**String toLowerCase()**](http://www.tutorialspoint.com/java/java_string_tolowercase.htm)  Converts all of the characters in this String to lower case using the rules of the default locale. | **x.toLowerCase();**  **String res = x.toLowerCase();**  **System.out.println(res); System.out.println("abc".toLowerCase());** |
| 11 | [**String toUpperCase()**](http://www.tutorialspoint.com/java/java_string_touppercase.htm)  Converts all of the characters in this String to upper case using the rules of the default locale. | **x.toLowerCase();**  **String res = x.toUpperCase();**  **System.out.println(res); System.out.println("abc".toUpperCase());** |
| 12 | [**String trim()**](http://www.tutorialspoint.com/java/java_string_trim.htm)  Returns a copy of the string, with leading and trailing whitespace omitted. | **String temp = " Hello world ";**  **System.out.println(temp.trim() ) ;**  **System.out.print(x.trim());** |

|  |  |
| --- | --- |
| Problem Statement | |
|  | You are playing a game where you must traverse a rectangular grid of cells using a spiral path. The map is given in a String[] **levelMap**, where the j-th character of the i-th element is the number of points associated with the cell in row i, column j. Rows are numbered from top to bottom, starting at 0, and columns are numbered from left to right, starting at 0. All coordinates in this problem will be given as (row, column). You start at cell (0,0), the top left corner of the grid. You are facing right. You move by repeating the following strategy until you have visited every single cell on the grid exactly once. If there is an adjacent cell in front of you that you haven't visited yet, move forward to that cell. Otherwise, if there are still unvisited cells on the grid, turn 90 degrees clockwise. To calculate your final score, add up all the points for the cells that you visited, but don't include the cells in which you changed direction. The first and last cells in your path will always be included in your final score. See examples for further clarification. |
| Definition | |
|  | |  |  | | --- | --- | | Class: | SpiralWalking | | Method: | totalPoints | | Parameters: | String[] | | Returns: | int | | Method signature: | int totalPoints(String[] levelMap) | | (be sure your method is public) | | |
| Limits | |
|  | |  |  | | --- | --- | | Time limit (s): | 2.000 | | Memory limit (MB): | 64 | |
| Constraints | |
| - | **levelMap** will contain between 2 and 50 elements, inclusive. |
| - | All elements of **levelMap** will contain the same number of characters. |
| - | Each element of **levelMap** will contain between 2 and 50 digits ('0'-'9'), inclusive. |
| Examples | |
| 0) |  |
|  | |  |  | | --- | --- | | |  | | --- | | {"111",  "111",  "111"} | | | Returns: 5 | | |  | | --- | | This is the spiral path you must follow: (0,0) -> (0,1) -> (0,2) -> (1,2) -> (2,2) -> (2,1) -> (2,0) -> (1,0) -> (1,1). | | |
| 1) |  |
|  | |  |  | | --- | --- | | |  | | --- | | {"101",  "110"} | | | Returns: 3 | | |  | | --- | | The grid is not always a square. | | |
| 2) |  |
|  | |  |  | | --- | --- | | |  | | --- | | {"00",  "10"} | | | Returns: 1 | | |  | | --- | |  | | |
| 3) |  |
|  | |  |  | | --- | --- | | |  | | --- | | {"86850",  "76439",  "15863",  "24568",  "45679",  "71452",  "05483"} | | | Returns: 142 | | |  | | --- | | The following image shows your path. The yellow cell is the last cell you visit. You receive points for all the cells except the red ones. http://www.topcoder.com/contest/problem/SpiralWalking/SW.gif | | |

This problem statement is the exclusive and proprietary property of TopCoder, Inc. Any unauthorized use or reproduction of this information without the prior written consent of TopCoder, Inc. is strictly prohibited. (c)2003, TopCoder, Inc. All rights reserved.