Computer Programming Assignment 2

Checkpoints

1. You should do the assignment in your own. You are not allowed to share code with others and/or copy code from other resources. If you are caught, as in the syllabus, you will get a failing grade.
2. Grading will be done in the Linux environment using Java 10.
3. Program failed to compile/run will result 0.
4. Do not loop your program to repeat unless you are told so.
5. Do not change input/output format unless you are told so.
6. Write your name and student number at top of program as a comment.
7. Do not include Korean (and any other language than English) comment. In some encoding formats, Korean comments will cause compilation errors in the Linux environment, which will result in a 0 for your grade.

Submission

1. Submit your assignment on eTL.
2. Zip your file (or tar) as '<Student ID>-assign2.zip'
    a. ex.) 2017-12345-assign2.zip
3. Due date of this assignment is Nov 1st, 2018
4. No late submission is allowed.

Problem 1 Reconstruction of some String methods

Notice: Do NOT use java’s String, Vector API for each methods.

Reconstruct following (public)methods of String by using char array.
All those methods will be members of “MyString”.

class name : MyString (MyString.java must be included in your assignment)

char[] toCharArray();
  Description: Convert MyString to char[] and return the char array.

boolean equals(MyString str);
  Description: Check whether all characters in the same index are the same or not.

boolean equalsIgnoreCase(MyString str);
  Description: Check whether all characters in the same index are the same or not regardless of Cases.

boolean startsWith(MyString str);
  Description: Check whether MyString starts with the sequence of characters in “str”.

boolean endsWith(MyString str);
  Description: Check whether MyString ends with the sequence of characters in “str”.

boolean contains(MyString str);
  Description: Check whether MyString contains sequence of characters in “str”.

int indexOf(MyString str);
  Description: Check whether MyString contains sequence of characters in “str”.

int length();
  Description: Returns total length of MyString.

MyString substring(int index1, int index2);
  Description: Returns the sequence of characters located in [index1, index2)

i.e. index1 <= i < index2.

MyString substring(int index1);

Description: Returns the suffix of MyString starting at index *index1*
i.e. index1 <= i <= end_Index.
Notice: end_Index does not indicate the location of null character included in all String objects.

MyString toLowerCase();

Description: Return MyString object whose characters are converted to lower cases.

MyString toUpperCase();

Description: Return MyString object whose characters are converted to upper cases.

Grading

(current directory: your_ID_assign2/)

```
javac *.java                //Grader.java  will be copied into your directory
java Grader                 //Grader.java will assume all methods are specified.
```

Problem 2 Tic Tac Toe 3D simulation

Let's implement a Tic Tac Toe game. You cannot use a game engine to create this game. The rules of the game is similar to the traditional Tic Tac Toe game. The rules are as follows: The object of Tic Tac Toe is to get three in a row. You play on a three by three game board. The first player is known as X and the second is O. Players alternate placing Xs and Os on the game board until either opponent has three in a row or all nine squares are filled. X always goes first, and in the event that no one has three in a row, the stalemate is called a cat game.

(Notice: the center of the middle board is unavailable, and invalid input should be checked in this problem)

Also, make sure to consider cases for a stalemate. If there ends up being no winner, print "The game is a tie. There is no winner."

So for example, the result below would make player X the winner, since there is a diagonal line of 3 Xs made.

```
+---+---+---+
| X | O | O |
+---+---+---+
| O | X | X |
+---+---+---+
| X | O | X |
+---+---+---+
```

However, we are going to put a slight variation to this game by making a 3x3x3 board. The rules are similar to traditional Tic Tac Toe. However, players can connect 3 in any 3 dimensional method allowed. Also, make sure to make the center panel of the Mid board unavailable for both players.

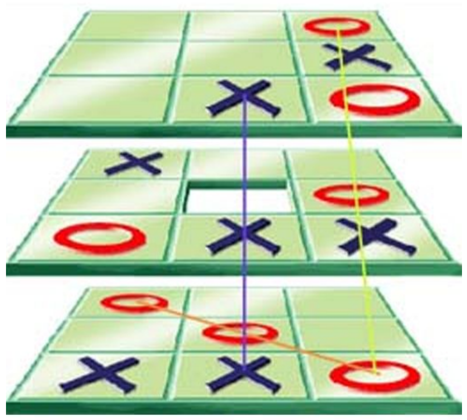
So for example, the following would make the X player the winner:

```
Top:

+---+---+---+
| o | x |   |
+---+---+---+
| o |   |   |
+---+---+---+
|   |   |   |
+---+---+---+

Mid:

+---+---+---+
|   | x |   |
+---+---+---+
|   | - |   |
+---+---+---+
|   |   |   |
+---+---+---+
```

```
Bot:
```

```
+---+---+---+
|   | X |   |
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
```

Here is a better representation of what we are trying to implement. It shows what kind of combinations are possible to reach a winner. All lines (Yellow, Blue, Orange) are valid win conditions.



Example

Output:

Enter Input for Player X:

Input : **T 1 1** // 1 1 = left down element

Output:

```
Top
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
| x |   |   |
+---+---+---+

Mid
+---+---+---+
|   |   |   |
+---+---+---+
|   | - |   |
+---+---+---+
|   |   |   |
+---+---+---+

Bot
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
```

Enter Input for Player O:

Input : **M 1 1**

Output:

```
Top
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
| x |   |   |
+---+---+---+

Mid:
+---+---+---+
|   |   |   |
+---+---+---+
|   | - |   |
+---+---+---+
| o |   |   |
+---+---+---+

Bot
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
```

Enter Input for Player X:

Input : **T 2 2**

Output:

```
Top
+---+---+---+
|   |   |   |
+---+---+---+
|   | x |   |
+---+---+---+
| x |   |   |
+---+---+---+

Mid
+---+---+---+
|   |   |   |
+---+---+---+
|   | - |   |
+---+---+---+
| o |   |   |
+---+---+---+
```

```
Bot
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
```

Enter Input for Player O:

Input : **M 1 2**

Output:

```
Top
+---+---+---+
|   |   |   |
+---+---+---+
|   | x |   |
+---+---+---+
| x |   |   |
+---+---+---+

Mid
+---+---+---+
|   |   |   |
+---+---+---+
| o | - |   |
+---+---+---+
| o |   |   |
+---+---+---+

Bot
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
```

Enter Input for Player X:

Input : **T 3 3**

Output:

```
Top
+---+---+---+
|   |   | x |
+---+---+---+
|   | x |   |
+---+---+---+
| x |   |   |
+---+---+---+

Mid
+---+---+---+
|   |   |   |
+---+---+---+
| o | - |   |
+---+---+---+
| o |   |   |
+---+---+---+

Bot
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+

Player 1 win!
```

Grading

(current directory: your_ID_assign2/)

```
javac TTT_3D.java
java TTT_3D
```