Tic Tac Toe

**Better work with 2 dimensional array of chars

Stage 1/5: Welcome to the battlefield! (medium)

Description

Tic-tac-toe is known all over the world. Each country may have its own version of the name, sometimes the rules are different, but the essence of the game remains the same. Below are the main rules that may be familiar to you since childhood.

Tic-tac-toe is a game played by two players on a 3x3 grid. One of the players is 'X', and the other player is 'O'. X plays first, then O takes the next turn, and so on.

The players write 'X' and 'O' on a 3x3 field.

The first player that puts 3 X's or 3 O's in a straight line (including diagonals) wins the game.

Objectives

Your first task in this project is to print the game grid in the console output. Use what you've learned about the print() function to print three lines, each containing three characters (X's and O's) to represent a game of tic-tac-toe where all fields of the grid have been filled in.

Example

The example below shows how your output might look.

```
X O X
O X O
X X O
```

Stage 2/5: The user is the gamemaster (hard)

Description

Our program should be able to display the grid at all stages of the game. Now we're going to write a program that allows the user to enter a string representing the game state and correctly prints the 3x3 game grid based on this input. We'll also add some boundaries around the game grid.

In this stage, you will write a program that:

Reads a string of 9 symbols from the input and displays them to the user in a 3x3 grid. The grid can contain only `X`, `O` and `_` symbols.

Outputs a line of dashes `---------` above and below the grid, adds a pipe `|` symbol to the beginning and end of each line of the grid, and adds a space between all characters in the grid.

Examples

Examples below show how your output should look.

Example 1:

```
Enter cells: O_OXXO_XX
---------
| O _ O |
| X X O |
| _ X X |
---------
```

Example 2:

```
Enter cells: OXO__X_OX
---------
| O X O |
| _ _ X |
| _ O X |
---------
```

Example 3:

```
Enter cells: _XO__X___
---------
| _ X O |
| _ _ X |
| _ _ _ |
---------
```

Stage 3/5: What's up on the field? (hard)

Description

In this stage, we're going to analyze the game state to determine if either of the players has already won the game or it is still ongoing, if the game is a draw, or if the user has entered an impossible game state (two winners, or with one player having made too many moves).

Objectives

In this stage, your program should:

1. Take a string entered by the user and print the game grid as in the previous stage.
2. Analyze the game state and print the result. Possible states:
    - **Game not finished** when neither side has three in a row but the grid still has empty cells.
    - **Draw** when no side has a three in a row and the grid has no empty cells.
    - **X wins** when the grid has three X's in a row.
    - **O wins** when the grid has three O's in a row.
    - **Impossible** when the grid has three X's in a row as well as three O's in a row, or there are a lot more X's than O's or vice versa (the difference should be 1 or 0; if the difference is 2 or more, then the game state is impossible).

In this stage, we will assume that either X or O can start the game.

Examples

The examples below show outputs and analysis results for different game states. Your program should work in the same way.

Example 1:	**Example 2**:	**Example 3**:
Enter cells: XXXOO__O_ --------- | X X X | | O O _ | | _ O _ | --------- X wins	Enter cells: XOXOXOXXO --------- | X O X | | O X O | | X X O | --------- X wins	Enter cells: XOOOXOXXO --------- | X O O | | O X O | | X X O | --------- O wins

Example 4: Enter cells: XOXOOXXXO --------- | X 0 X | | 0 0 X | | X X 0 | --------- Draw	**Example 5**: Enter cells: XO_OOX_X_ --------- | X 0 _ | | 0 0 X | | _ X _ | --------- Game not finished	**Example 6**: Enter cells: XO_XO_XOX --------- | X 0 _ | | X 0 _ | | X 0 X | --------- Impossible
Example 7: Enter cells: _O_X__X_X --------- | _ 0 _ | | X _ _ | | X _ X | --------- Impossible	**Example 8**: Enter cells: _OOOO_X_X --------- | _ 0 0 | | 0 0 _ | | X _ X | --------- Impossible	

Stage 4/5: First move! (hard)

Description

It's time to make our game interactive! Now we're going to add the ability for a user to make a move.

To do this, we need to divide the grid into cells.

Suppose the top left cell has the coordinates `(1, 1)` and the bottom right cell has the coordinates `(3, 3)` like in this table:

```
(1, 1) (1, 2) (1, 3)
(2, 1) (2, 2) (2, 3)
(3, 1) (3, 2) (3, 3)
```

The program should ask the user to enter the coordinates of the cell where they want to make a move.

In this stage, the user plays as `X`, not `O`. Keep in mind that the first coordinate goes from top to bottom and the second coordinate goes from left to right. Also note that coordinates start with 1 and can be 1, 2, or 3.

What happens if the user enters incorrect coordinates? The user could enter symbols instead of numbers, or enter coordinates representing occupied cells or cells that aren't even on the grid. You need to check the user's input and handle possible errors.

Objectives

The program should work as follows:

- Get the 3x3 grid from the input as in the previous stages.
- Output this 3x3 grid as in the previous stages.
- Prompt the user to make a move.
- The user should input 2 numbers that represent the cell where they want to place their X. (the 9 symbols representing the field will be the first line of input, and the 2 coordinate numbers will be the second line of input)
- Analyze user input and show messages in the following situations:
    - **This cell is occupied! Choose another one!** if the cell is not empty.

- **You should enter numbers!** if the user enters non-numeric symbols in the coordinates input. (**Hint: to check if the char is a number use the static method `isDigit` from `Character` class. e.g. `Character.isDigit('0')`**)
- **Coordinates should be from 1 to 3!** if the user enters coordinates outside the game grid.

- Update the grid to include the user's move and print the updated grid to the console.

The program should also check the user's input. If the input is unsuitable, the program should tell the user why their input was wrong, and prompt them to enter coordinates again.

To summarize, you need to output the game grid based on the first line of input, and then ask the user to enter a move. Keep asking until the user enters coordinates that represent an empty cell on the grid, update the grid to include that move, and then output it to the console. You should output the field only 2 times: once before the user's move, and once after the user has entered a legal move.

Do not delete the code you already wrote that analyzes the game state; you will need it in the final step of this project.

Example

Example 1:	**Example 2:**	**Example 3:**
Enter cells: X_X_O____ --------- | X _ X | | _ O _ | | _ _ _ | --------- **Enter the coordinates**: 3 1 --------- | X _ X | | _ O _ | | X _ _ | ---------	**Enter cells**: _XXOO_OX_ --------- | _ X X | | O O _ | | O X _ | --------- **Enter the coordinates:** 1 1 --------- | X X X | | O O _ | | O X _ | ---------	**Enter cells:** _XXOO_OX_ --------- | _ X X | | O O _ | | O X _ | --------- **Enter the coordinates:** 3 3 --------- | _ X X | | O O _ | | O X X | ---------

Example 4:
Enter cells: _XXOO_OX_
```
---------
| _ X X |
| 0 0 _ |
| 0 X _ |
---------
```
Enter the coordinates: 2 3
```
---------
| _ X X |
| 0 0 X |
| 0 X _ |
---------
```

Example 5:
Enter cells: _XXOO_OX_
```
---------
| _ X X |
| 0 0 _ |
| 0 X _ |
---------
```
Enter the coordinates: 3 1
This cell is occupied! Choose another one!
Enter the coordinates: 1 1
```
---------
| X X X |
| 0 0 _ |
| 0 X _ |
---------
```

Example 4:
Enter cells: _XXOO_OX_
```
---------
|   X X |
| 0 0   |
| 0 X   |
---------
```
Enter the coordinates: 4 1
Coordinates should be from 1 to 3!
Enter the coordinates: 1 4
Coordinates should be from 1 to 3!
Enter the coordinates: 1 1
```
---------
| X X X |
| 0 0   |
| 0 X   |
---------
```

Example 6:
Enter cells: _XXOO_OX_
```
---------
| _ X X |
| 0 0 _ |
| 0 X _ |
---------
```
Enter the coordinates: one
You should enter numbers!
Enter the coordinates: one one
You should enter numbers!
Enter the coordinates: 1 1
```
---------
| X X X |
| 0 0 _ |
| 0 X _ |
---------
```

Stage 5/5: Fight! (hard)

Description

Our game is almost ready! Now let's combine what we've learned in the previous stages to make a game of tic-tac-toe that two players can play from the beginning (with an empty grid) through to the end (until there is a draw, or one of the players wins).

The first player has to play as X and their opponent plays as O.

Objectives

In this stage, you should write a program that:

Prints an empty grid at the beginning of the game.

Creates a game loop where the program asks the user to enter the cell coordinates, analyzes the move for correctness and shows a grid with the changes if everything is okay.

Ends the game when someone wins or there is a draw.

You need to output the final result at the end of the game.

Good luck!

The project was changed. Now the coordinates start from the upper left corner. Look closely at the examples.

Example

The example below shows how your program should work.

```
---------
| _ _ _ |
| _ _ _ |
| _ _ _ |
---------
```

Enter the coordinates: 2 2

```
---------
| _ _ _ |
| _ X _ |
| _ _ _ |
---------
```

Enter the coordinates: 2 2

This cell is occupied! Choose another one!

Enter the coordinates: two two

You should enter numbers!
Enter the coordinates: 1 4
Coordinates should be from 1 to 3!
Enter the coordinates: 1 1

```
---------
| O _ _ |
| _ X _ |
| _ _ _ |
---------
```

Enter the coordinates: 3 3

```
---------
| O _ _ |
| _ X _ |
| _ _ X |
---------
```

Enter the coordinates: 2 1

```
---------
| O _ _ |
| O X _ |
| _ _ X |
---------
```

Enter the coordinates: 3 1

```
---------
| O _ _ |
| O X _ |
| X _ X |
---------
```

Enter the coordinates: 2 3

```
---------
| O _ _ |
| O X O |
| X _ X |
---------
```

Enter the coordinates: 3 2

```
---------
| O _ _ |
| O X O |
| X X X |
---------
```

X wins