

CSCI-101 Programming 1

Week of Oct 9 - Lab 8c

INSTRUCTIONS

Include the files for this lab in the **lab8** directory.

The following program uses a 10x10 array of characters as a game field. The user's current location in the game field is indicated by the 'X' character. The program for this lab simply allows the user to move the 'X' character around the field.

When done, expand the program on your own to include mines, walls and other game elements.

Create a file named **Lab8c.java** and include the following:

1. Add a method named **initializePlayingField**. The method has one parameter which holds a reference to a 2D array of characters.

The method sets the last element in the last row to 'X' and all other elements in the 2D array to the '=' character.

2. Add a method named **printField**. The method has a single parameter that holds a reference to a 2D array of characters. The method prints the elements in the matrix to the screen, with each row on a separate line, and with the elements in a row separated by spaces.

3. Add a method named **isValidDirection**. The method has 3 parameters. The first parameter is named **row** and holds an integer, the second is named **col** and holds an integer, and the third is named **direction** and holds a character.

The variables named **row** and **col** hold indices specifying the player's current location in a (10 x 10) 2D array.

The value in the variable named **direction** is 'w' (up), 'a' (left), 's' (down) or 'd' (right) indicating where the player would like to move.

If the location of the player's next move doesn't take the player out of bounds in the 2D array return true; Otherwise return false.

4. Add a method named **updateField**. The method has four parameters. The first parameter is named **matrix** and holds a reference to a 2D array of characters, the second parameter is named **row** and holds an integer, the third parameter is named **col** and holds an integer, and the fourth parameter is named **direction** and holds a character.

The variables named **row** and **col** hold indices specifying the player's current location in **matrix**.

The method moves the 'X' character in **matrix** in the direction specified in the variable named **direction**, replacing the 'X' character at the user's current location with a '=' character.

5. In the **main** method do the following.
 1. Declare the following variables:

```
int playerRow = 9;  
int playerCol = 9;  
String input = null;  
char choice = '?';  
boolean isValid = false;
```

2. Create a Scanner that can read data from the keyboard.
3. Declare a (10 x 10) 2D array of characters named **field**.
4. Initialize the 2D array named **field** by calling **initializePlayingField**.
5. Add a do-while loop that executes a block of code while the value in the variable named **choice** is not 'x'.

```
do {  
  
} while (choice != 'x');
```

6. Inside the **do-while** loop do the following:
 - a. Print the game field by calling **printField**.
 - b. Ask the user to enter a direction: w (up), a (left), s (down), d (right) or x (to exit)
 - c. Read the user's choice from the keyboard and store the value in the variable named **input**.
 - d. If the length of the string entered by the user is 0, set **isValid** to false; Otherwise set **choice** equal to the first character in the string entered by the user and set **isValid** equal to the value returned by **isValidDirection**.
 - e. If the direction is valid, call **updateField** and update the values in **playerRow** and **playerCol**.
 - f. Reset the screen using the following code:

```
System.out.print("\033[H\033[2J");
System.out.flush();
```