CSCI-101 Programming 1

Lab 9

INSTRUCTIONS

Inside your **lab9** directory create a file named **Lab9.java** and include in it code that satisfies the following.

The following program uses a 10x10 array of characters as a game field. The users 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.

Inside the **Lab9.java** file write a program that satisfies 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 first 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 either 'w' (up), 'a' (left), 's' (down) or 'd' (right) indicating where the player would like to move.

   If the value in direction is not 'w', 'a', 's', or 'd' then return **false**.

   If the direction the player would like to move takes the player out of bounds return **false**; otherwise return **true**.

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 the direction specified in the variable named direction and replaces the 'X' character at the user's current location with an '=' character.

5. In the **main** method do the following.

    1. Create a Scanner that can read data from the keyboard.

    2. Declare a (10 x 10) 2D array of characters named **field**.

    3. Initialize the 2D array by calling **initializePlayingField**.

    4. Declare the following variables:

```
int playerRow = 9;          // the row of the player's current position
int playerCol = 0;          // the column of the player's current position
String input = null;        // holds the string read from the keyboard
char choice = '?';          // the direction the player wishes to go
boolean isValid = false;    // true if the direction the player chose is a valid direction
```

    5. Add an infinite loop that does 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 then print a message to the user and **continue** to the top of the loop.

        e. Set **choice** equal to the first character in the string entered by the user.

        f. If the user enters 'x' to exit, terminate the program.

        g. Set **isValid** equal to the value returned by **isValidDirection()**.

        h. If the direction is valid, call **updateField()** and update the values in **playerRow** and **playerCol** to indicate the players new current position.

        i. Reset the screen using the following code:

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