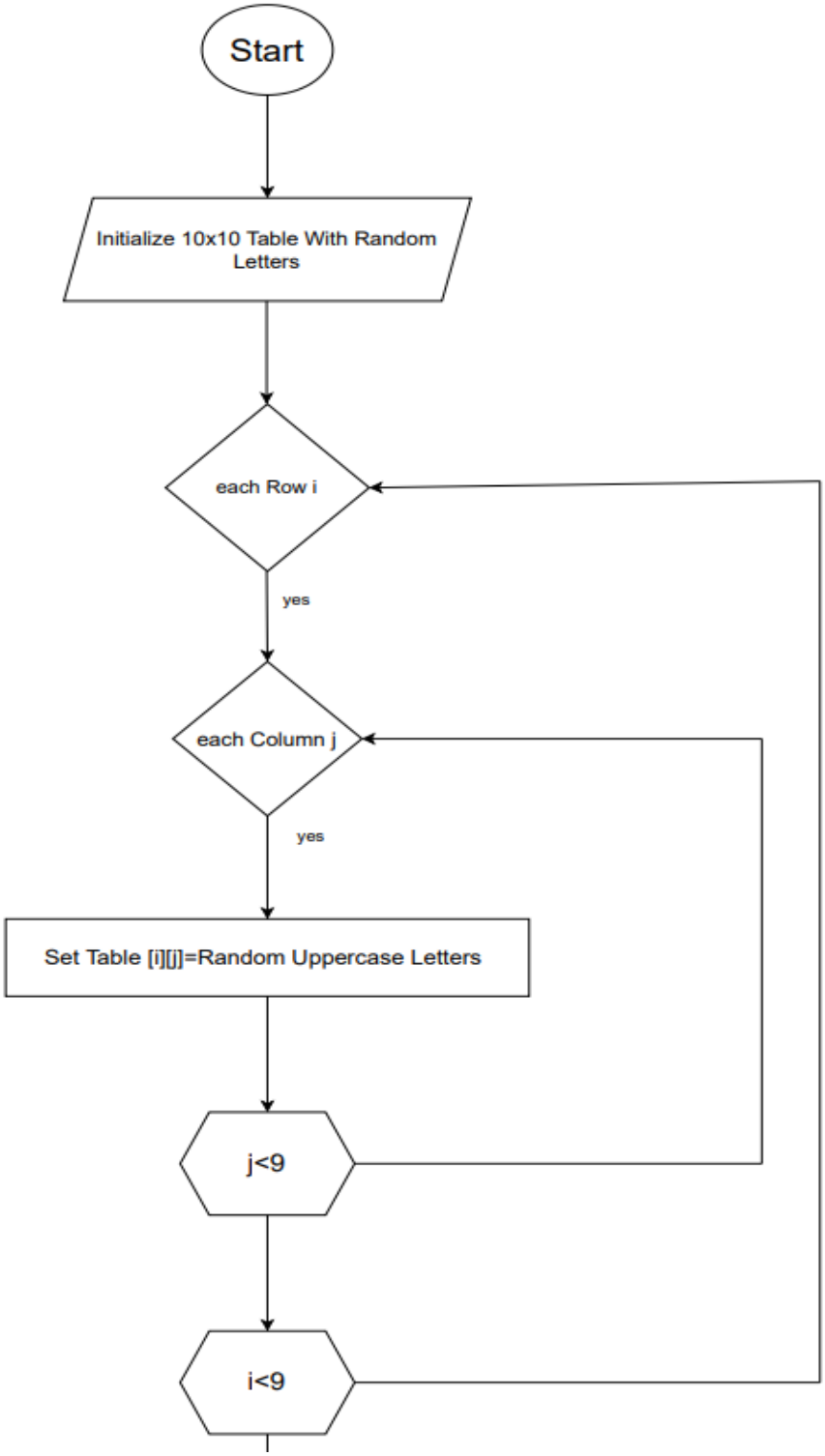
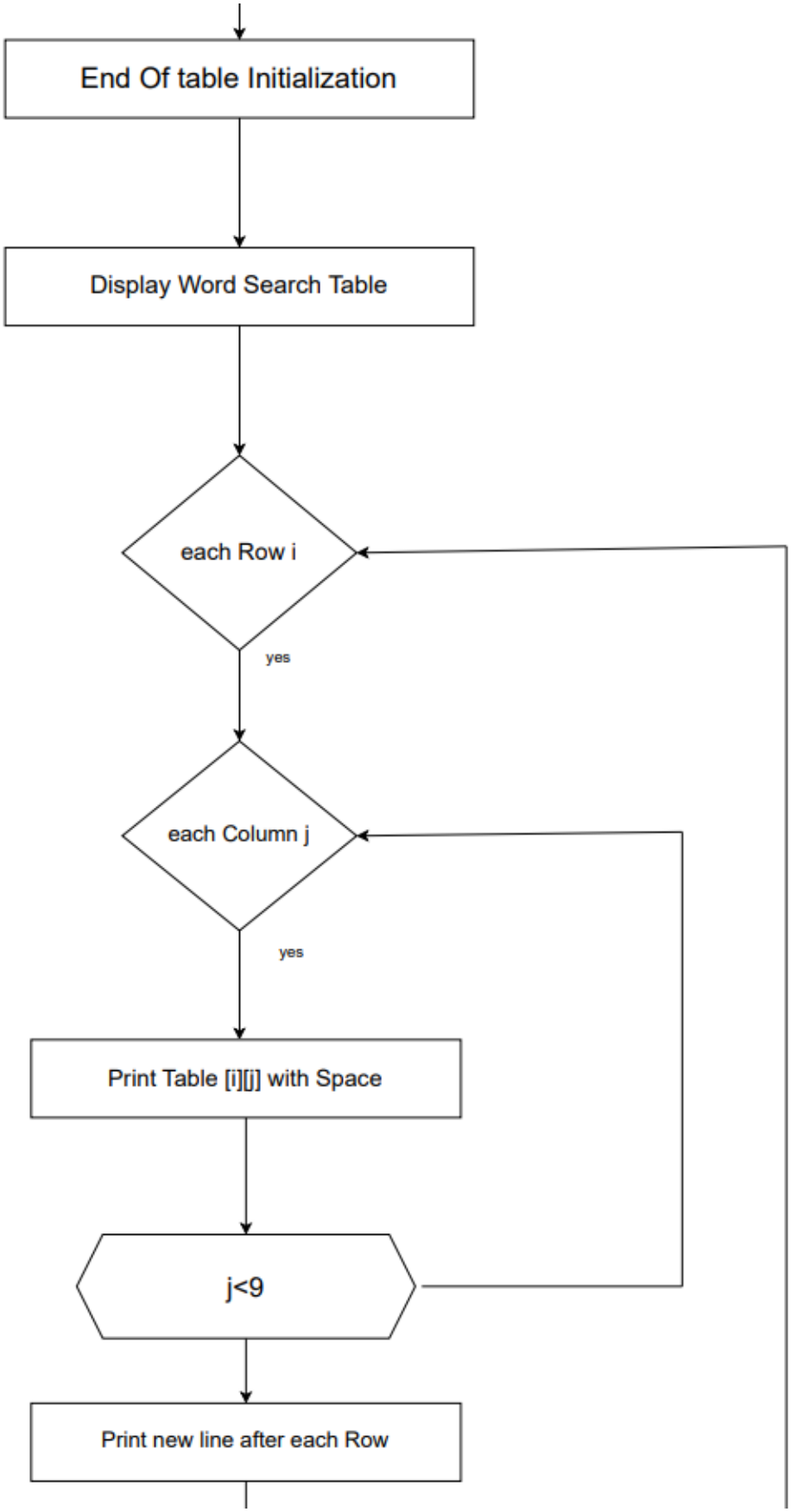
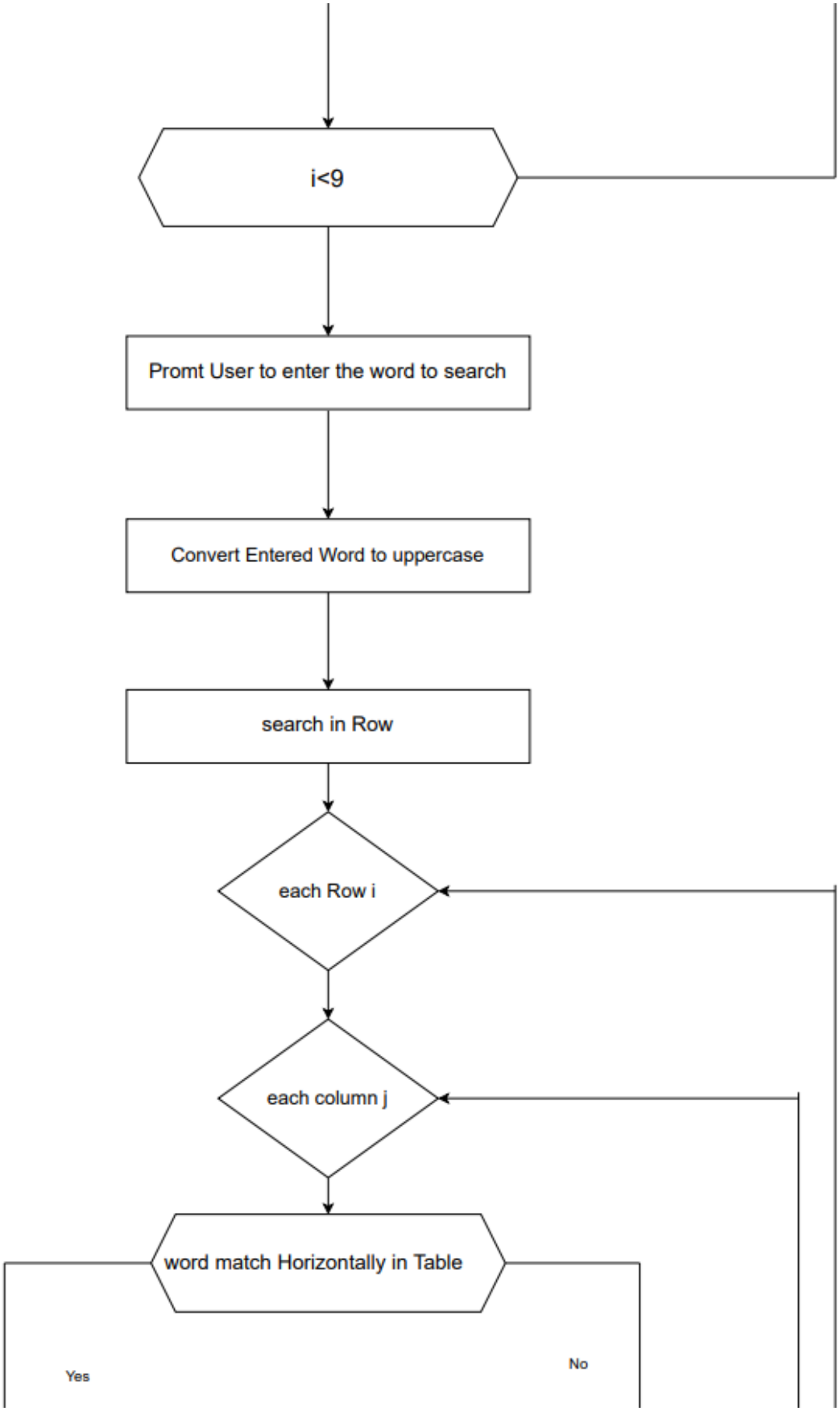
1.UML Diagram

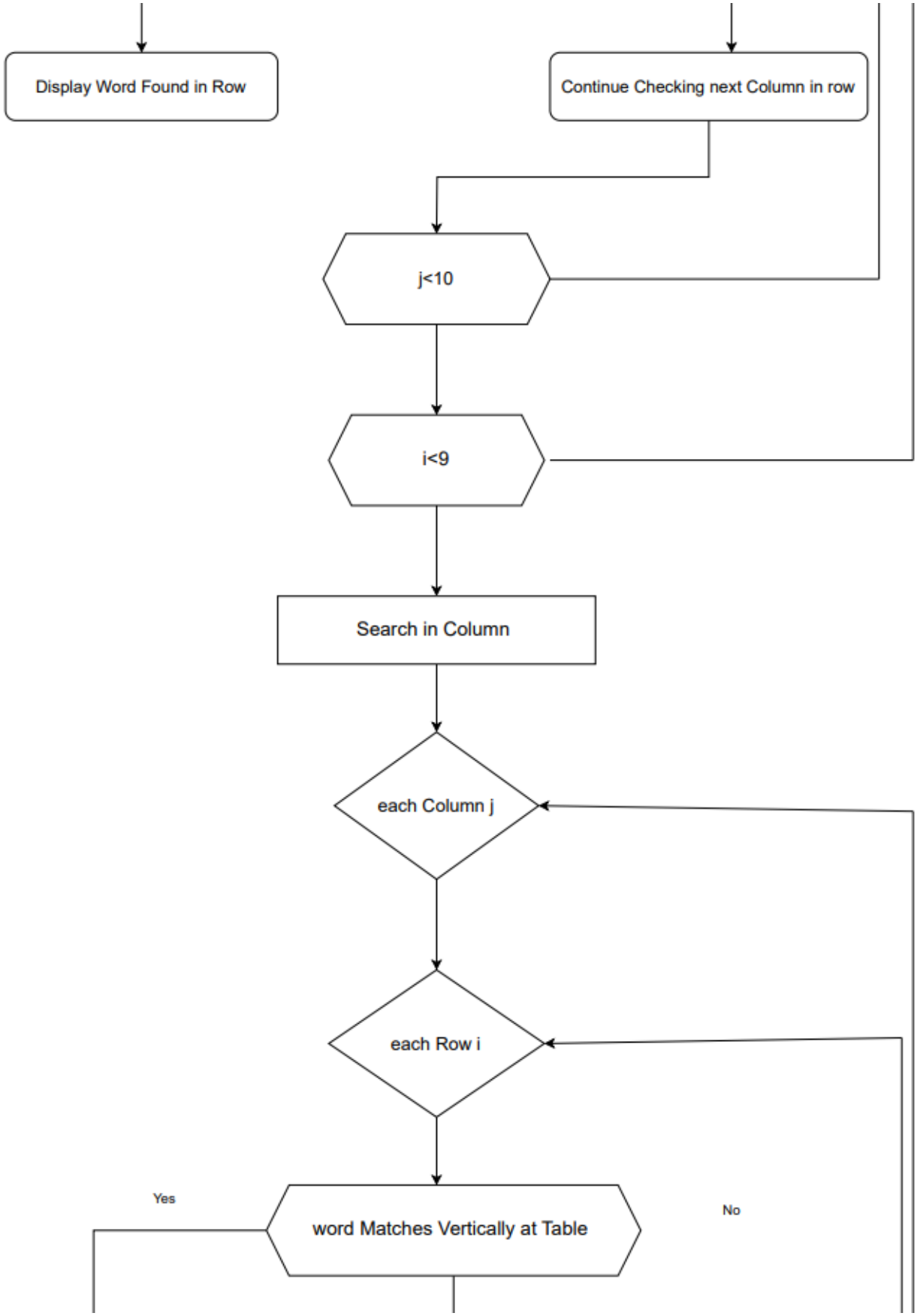
Word_Search
- SIZE : int -table : char[][]
+ main(args : String[]) - initializeTable() - displayTable() - findWord(word : String) : boolean - checkRight(row : int, col : int, word : String) : boolean - checkDown(row : int, col : int, word : String) : boolean

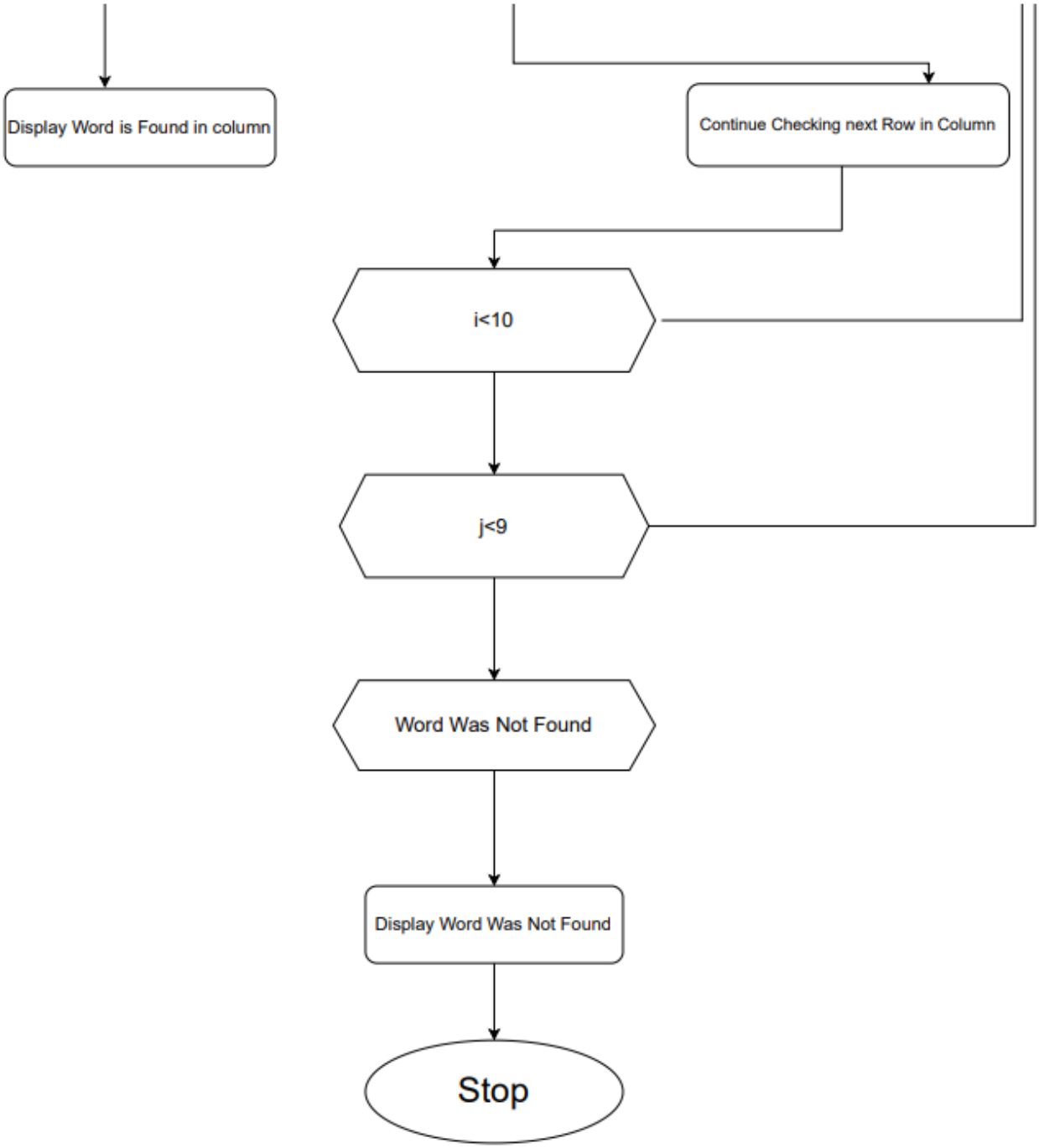
2.Flowchart


Start
Initialize 10x10 Table With Random Letters
each Row i
yes
each Column j
yes
Set Table [i][j]=Random Uppercase Letters
j<9
i<9


End Of table Initialization
Display Word Search Table
each Row i
yes
each Column j
yes
Print Table [i][j] with Space
j<9
Print new line after each Row


i<9
Promt User to enter the word to search
Convert Entered Word to uppercase
search in Row
each Row i
each column j
word match Horizontally in Table
Yes
No


Display Word Found in Row
Continue Checking next Column in row
j<10
i<9
Search in Column
each Column j
each Row i
Yes
word Matches Vertically at Table
No


Display Word is Found in column
Continue Checking next Row in Column
i<10
j<9
Word Was Not Found
Display Word Was Not Found
Stop

3.Code of World_Search Game

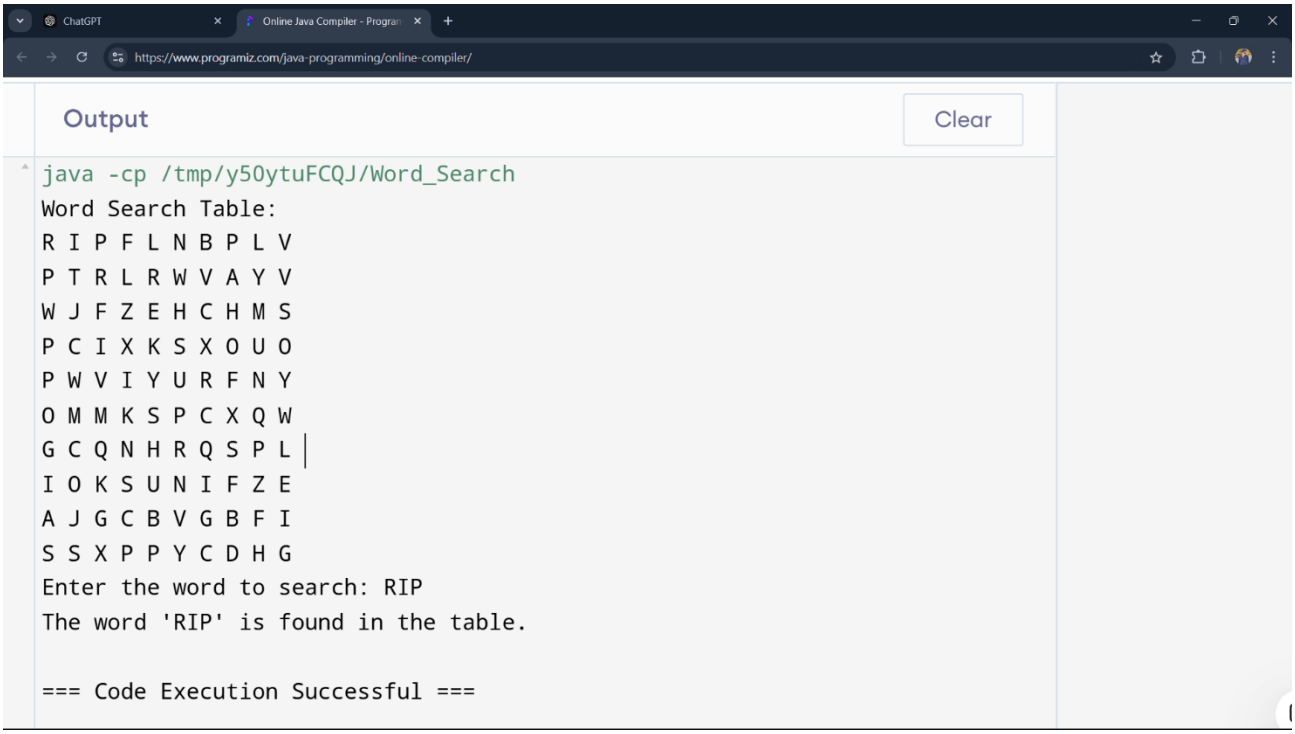
```
import java.util.Scanner;

public class Word_Search
{
    private static final int SIZE = 10;
    private static char[][] table = new char[SIZE][SIZE];

    // Method to initialize the table with random letters
    private static void initializeTable()
    {
        // Use a Random object to generate random characters
        java.util.Random random = new java.util.Random();

        // Fill the table with random uppercase letters
        for (int i = 0; i < SIZE; i++)
        {
            for (int j = 0; j < SIZE; j++)
            {
                // Generate a random uppercase letter (A-Z)
                table[i][j] = (char) ('A' + random.nextInt(26));
            }
        }
    }

    // Method to display the table
    private static void displayTable()
    {
```

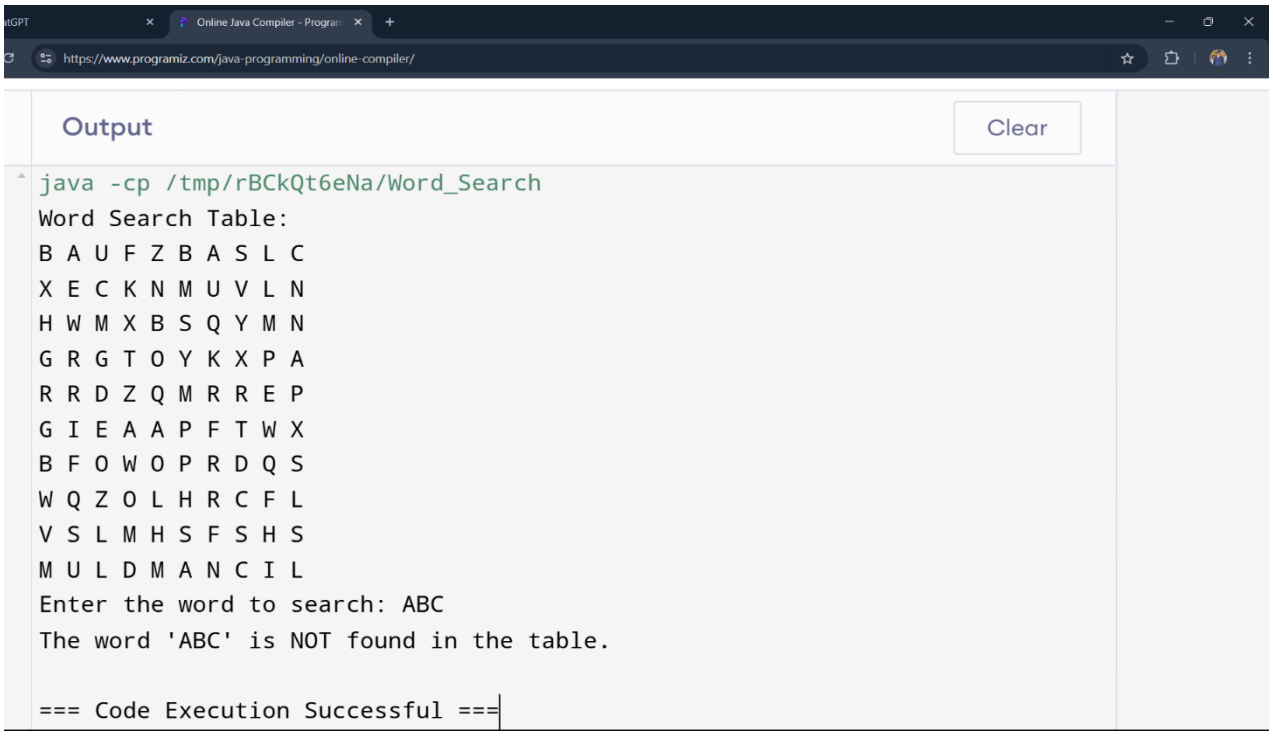
```
    for (int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j < SIZE; j++)
        {
            System.out.print(table[i][j] + " ");
        }
        System.out.println();
    }
}

// Check if a word is in the table
private static boolean findWord(String word)
{
    int length = word.length();

    // Check rows (horizontal)
    for (int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j <= SIZE - length; j++)
        {
            if (checkRight(i, j, word)) return true;
        }
    }

    // Check columns (vertical)
    for (int i = 0; i <= SIZE - length; i++)
    {
        for (int j = 0; j < SIZE; j++)
```

```
            {
                if (checkDown(i, j, word)) return true;
            }
        }

        return false; // word not found
    }

    // Check right (horizontal)
    private static boolean checkRight(int row, int col, String word)
    {
        for (int i = 0; i < word.length(); i++)
        {
            if (table[row][col + i] != word.charAt(i)) return false;
        }
        return true;
    }

    // Check down (vertical)
    private static boolean checkDown(int row, int col, String word)
    {
        for (int i = 0; i < word.length(); i++)
        {
            if (table[row + i][col] != word.charAt(i)) return false;
        }
        return true;
    }
```

```
    // Main method
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        initializeTable();

        System.out.println("Word Search Table:");
        displayTable();

        System.out.print("Enter the word to search: ");
        String word = scanner.nextLine().toUpperCase();

        if (findWord(word))
        {
            System.out.println("The word '" + word + "' is found in the table.");
        }
        else
        {
            System.out.println("The word '" + word + "' is NOT found in the table.");
        }

    }
}
```

4.OUTPUT


ChatGPT
Online Java Compiler - Progra
https://www.programiz.com/java-programming/online-compiler/
Output
Clear
java -cp /tmp/y5OytuFCQJ/Word_Search
Word Search Table:
R I P F L N B P L V
P T R L R W V A Y V
W J F Z E H C H M S
P C I X K S X O U O
P W V I Y U R F N Y
O M M K S P C X Q W
G C Q N H R Q S P L
I O K S U N I F Z E
A J G C B V G B F I
S S X P P Y C D H G
Enter the word to search: RIP
The word 'RIP' is found in the table.
=== Code Execution Successful ===


Online Java Compiler - Progra
https://www.programiz.com/java-programming/online-compiler/
Output
Clear
java -cp /tmp/rBCkQt6eNa/Word_Search
Word Search Table:
B A U F Z B A S L C
X E C K N M U V L N
H W M X B S Q Y M N
G R G T O Y K X P A
R R D Z Q M R R E P
G I E A A P F T W X
B F O W O P R D Q S
W Q Z O L H R C F L
V S L M H S F S H S
M U L D M A N C I L
Enter the word to search: ABC
The word 'ABC' is NOT found in the table.
=== Code Execution Successful ===

5.Explanation of code

1. Import the Scanner Class:

- `import java.util.Scanner;` is used to get input from the user.

2. Define the Class:

- `public class Word_Search` defines the class named `Word_Search`. This is the main structure that will hold all the code.

3. Declare Constants and Variables:

- `private static final int SIZE = 10;` sets a constant `SIZE` to 10, meaning the table will have 10 rows and 10 columns.
- `private static char[][] table = new char[SIZE][SIZE];` creates a 10x10 character array called `table` to hold letters.

4. Initialize the Table with Random Letters:

- `private static void initializeTable()` is a method that fills the table with random letters.
- Inside this method:
- `java.util.Random random = new java.util.Random();` creates a Random object to generate random numbers.
- A nested `for` loop goes through each cell in the `table`.
- `(char) ('A' + random.nextInt(26))` generates a random letter from 'A' to 'Z' and places it in each cell.

5. Display the Table:

- `private static void displayTable()` is a method that prints the `table` in a grid format.
- Inside this method:
- A nested `for` loop goes through each cell in `table`, printing each letter with a space in between.
- `System.out.println();` moves to a new line after each row to create the grid format.

6. Check if the Word is in the Table:

- `private static boolean findWord(String word)` checks if the user’s word is in the table.
- `int length = word.length();` gets the length of the word.

- Two `for` loops are used:
- The first loop checks horizontally (across each row).
- `checkRight(i, j, word)` calls another method to check if the word matches horizontally.
- The second loop checks vertically (down each column).
- `checkDown(i, j, word)` checks if the word matches vertically.
- If either method finds the word, `findWord` returns `true`.

7. Check Right (Horizontal):
   - `private static boolean checkRight(int row, int col, String word)` checks if a word matches from left to right in a row.
   - A `for` loop goes through each letter in `word` and compares it to letters in `table`.
   - If any letters do not match, it returns `false`. If all match, it returns `true`.

8. Check Down (Vertical):
   - `private static boolean checkDown(int row, int col, String word)` checks if a word matches from top to bottom in a column.
   - It works similarly to `checkRight`, but moves vertically.

9. Main Method:
   - `public static void main(String[] args)` is the entry point of the program.
   - Inside `main`:
   - `Scanner scanner = new Scanner(System.in);` creates a `Scanner` object to read user input.
   - `initializeTable();` fills the table with random letters.
   - `displayTable();` prints the random letters table to the screen.
   - The program then asks the user to enter a word to search: `System.out.print("Enter the word to search: ");`.
   - The user's input is read and converted to uppercase: `String word = scanner.nextLine().toUpperCase();`.
   - `findWord(word)` is called to check if the word is in the table.
   - If `findWord` returns `true`, a message shows the word was found. If `false`, a message says the word was not found.

CONCLUSION

This code creates a simple word search game in Java. It generates a 10x10 grid filled with random letters and allows the user to enter a word to search for. The program then checks if the word appears in the grid either horizontally or vertically and notifies the user if the word is found. This provides a straightforward example of using 2D arrays, random letter generation, and basic string search techniques in Java.