

## Level 2 Tasks:-

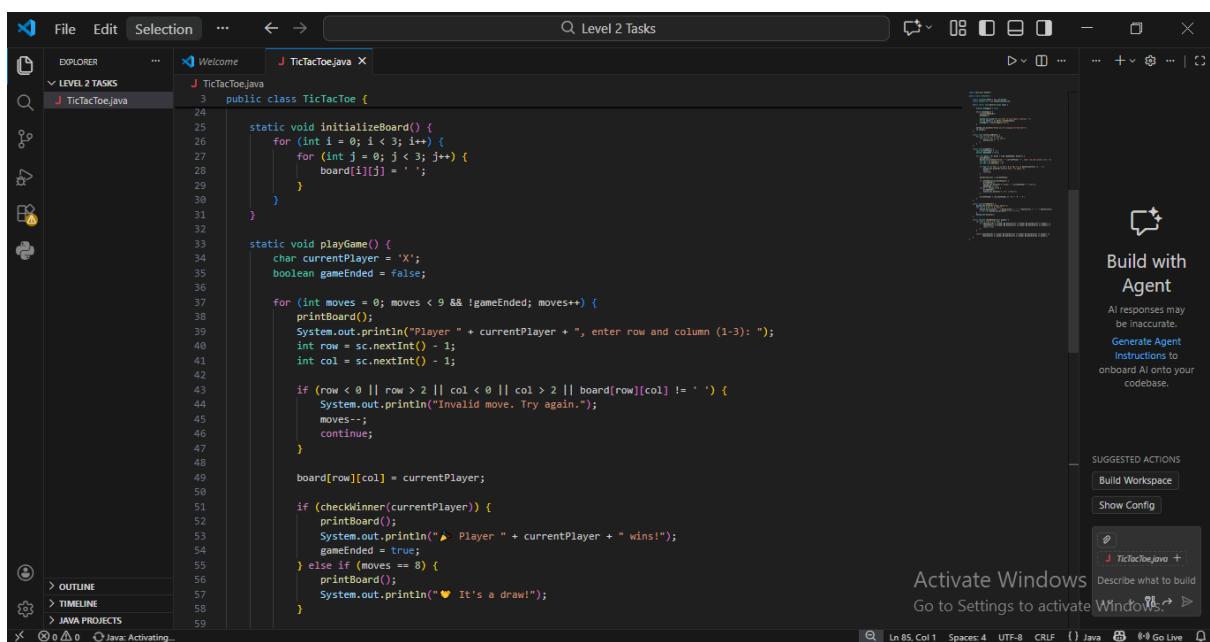
### Task 1

#### Task: Tic-Tac-Toe Game

Description: Implement a two-player tic-tac-toe game. Display the game board and prompt each player to enter their moves. Check for a winning condition or a draw after each move, and display the result accordingly. Allow the players to play multiple rounds if desired.

#### Skills:

Arrays or matrices, loops, conditional statements.



The screenshot shows a Java code editor interface with the following details:

- File Bar:** File, Edit, Selection, ...
- Title Bar:** Welcome J TicTacToe.java
- Explorer Bar:** LEVEL 2 TASKS, TicTacToe.java
- Code Area:** The code for the TicTacToe class is displayed. It includes methods for initializing the board, playing a game, and checking for a winner. The code uses nested loops for the board and conditional statements for player input and winning logic.
- Right Panel:**
  - Build with Agent:** A sidebar with instructions to onboard AI onto the codebase.
  - Suggested Actions:** Buttons for "Build Workspace" and "Show Config".
  - Activate Windows:** A message prompting to go to settings to activate Windows.
- Bottom Status Bar:** Line 85, Col 1, Spaces: 4, UTF-8, CRLF, Java, Go Live, etc.

The screenshot shows a Java development environment with the following details:

- File Explorer:** Shows files `TicTacToe.java` and `TicTacToe.class` under the folder `LEVEL 2 TASKS`.
- Code Editor:** Displays the `TicTacToe.java` file with code for initializing a 3x3 board and playing a game.
- Terminal:** Shows the command `javac TicTacToe.java` being run, followed by the output of the game being played.
- Output:** Shows the game board state and player interactions.
- Suggested Actions:** Includes "Build Workspace" and "Show Config".
- Right Panel:** Contains a "Build with Agent" section and an "Activate Windows" message.

```
public class TicTacToe {
    static void initializeBoard() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                board[i][j] = ' ';
            }
        }
    }

    static void playGame() {
        char currentPlayer = 'X';
        boolean gameEnded = false;

        for (int moves = 0; moves < 9 && !gameEnded; moves++) {
            printBoard();
            System.out.println("Player " + currentPlayer + ", enter row and column (1-3): ");
            int row = sc.nextInt() - 1;
            int col = sc.nextInt() - 1;

            if (row < 0 || row > 2 || col < 0 || col > 2) {
                System.out.println("Invalid input! Please enter valid row and column values (1-3).");
                continue;
            }

            if (board[row][col] != ' ') {
                System.out.println("Cell already occupied! Please choose an empty cell.");
                continue;
            }

            board[row][col] = currentPlayer;
            if (checkWin(row, col)) {
                gameEnded = true;
                System.out.println(currentPlayer + " wins!");
            } else if (moves == 8) {
                gameEnded = true;
                System.out.println("It's a draw!");
            }
        }
    }

    static void printBoard() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print(board[i][j] + " | ");
            }
            System.out.println();
        }
    }

    static boolean checkWin(int row, int col) {
        // Check row
        for (int i = 0; i < 3; i++) {
            if (board[i][col] != board[row][i]) {
                return false;
            }
        }
        // Check column
        for (int i = 0; i < 3; i++) {
            if (board[row][i] != board[i][col]) {
                return false;
            }
        }
        // Check diagonals
        if (row == col) {
            for (int i = 0; i < 3; i++) {
                if (board[i][i] != board[row][i]) {
                    return false;
                }
            }
        }
        if (row + col == 2) {
            for (int i = 0; i < 3; i++) {
                if (board[i][2 - i] != board[row][i]) {
                    return false;
                }
            }
        }
        return true;
    }
}
```

PS C:\Users\Aaryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks> cd "c:\Users\Aaryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks"> if (\$?) { java TicTacToe }> 1 2>

Game Board:  
X | X | X  
-----  
O
O

? Player X wins!  
Do you want to play again? (yes/no): no  
Thank you for playing Tic-Tac-Toe!

PS C:\Users\Aaryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks>

## Task2: Password Strength Checker

Description: Create a program that checks the strength of a password. Prompt the user to input a password and analyze its strength based on certain criteria, such as length, presence of uppercase letters, lowercase letters, numbers, and special characters. Provide feedback on the password strength.

Skills:

String statements.

The screenshot shows a Java code editor interface with the following details:

- File Explorer:** Shows files in the "LEVEL 2 TASKS" folder: `TicTacToe.java`, `TicTacToe.class`, and `PasswordStrengthChecker.java`.
- Code Editor:** Displays the `PasswordStrengthChecker.java` file content. The code reads a password from the user, calculates its strength based on length and character complexity, and prints a summary.
- Output Panel:** Shows the terminal output of the program's execution.
- Right Sidebar:** Includes a "Build with Agent" section with AI-related options and a "SUGGESTED ACTIONS" panel with "Build Workspace" and "Show Config" buttons.
- Bottom Status Bar:** Shows the current line (Ln 8, Col 1), selected text (2 selected), and other system information like spaces, encoding, and Java version.

This screenshot is similar to the first one but includes a "TERMINAL" tab at the bottom of the interface. The terminal shows the following session:

```
PS C:\Users\Aryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks> cd "c:\Users\Aryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks"
PS C:\Users\Aryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks> if ($?) { java PasswordStrengthChecker }
Enter a password:
Password Analysis:
Strength: WEAK
Suggestion: Use at least 8 characters, include uppercase, lowercase, numbers, and special symbols.
PS C:\Users\Aryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks> cd "c:\Users\Aryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks"
PS C:\Users\Aryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks> if ($?) { java PasswordStrengthChecker }
● Enter a password: aKearZ12$hs

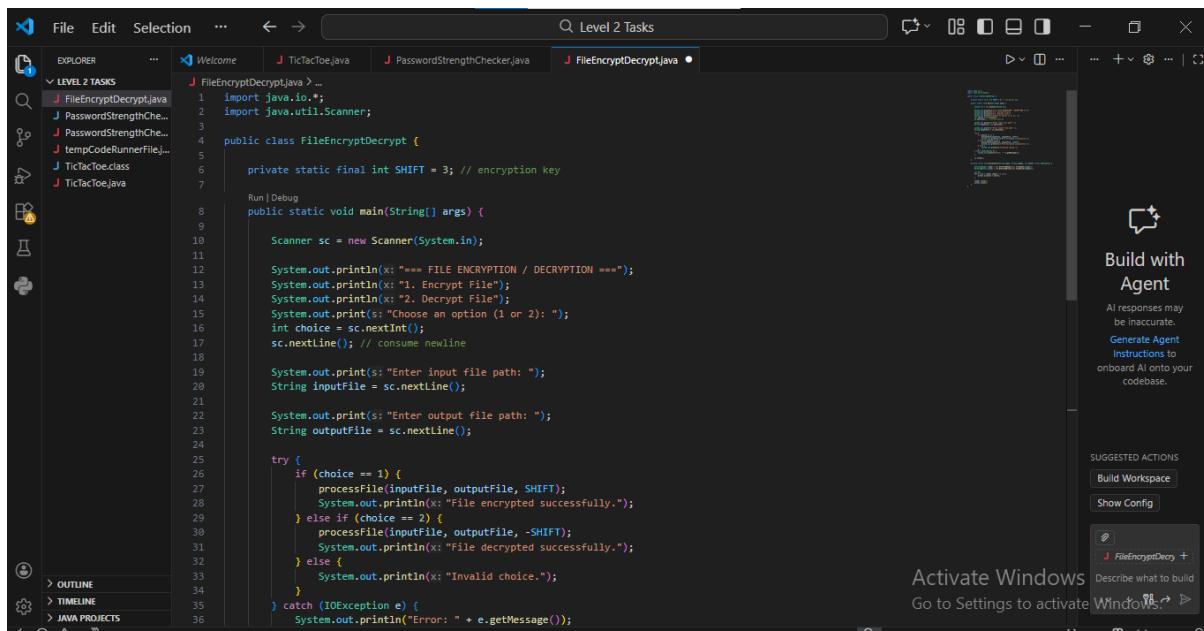
Password Analysis:
Strength: STRONG
Good job! Your password is secure.
PS C:\Users\Aryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks> cd "c:\Users\Aryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks"
PS C:\Users\Aryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks> if ($?) { java PasswordStrengthChecker }
● Enter a password: 
```

## Task3: File Encryption/Decryption

Description: Create a program that encrypts or decrypts the contents of a text file using a simple encryption algorithm. Prompt the user to choose between encryption or decryption, and input the file name or path. Encrypt or decrypt the file accordingly and save the result to a new file.

Skills:

File handling, string manipulation, basic input/output operations.



The screenshot shows a Java development environment with the following details:

- File Explorer:** Shows files including `FileEncryptDecrypt.java`, `TicTacToe.java`, `TempCodeRunnerFile.java`, and `TicTacToe.class`.
- Code Editor:** The active tab is `FileEncryptDecrypt.java`. The code implements a simple encryption/decryption algorithm using a key of 3. It prompts the user for an option (1 for Encrypt, 2 for Decrypt) and file paths, then processes the file accordingly.
- Right Panel:** Includes a "Build with Agent" section with AI-related instructions and a "SUGGESTED ACTIONS" panel with options like "Build Workspace" and "Show Config".
- Bottom Status Bar:** Displays "Activate Windows" and "Go to Settings to activate Windows".

```
File Encrypt Decrypt
import java.io.*;
import java.util.Scanner;
public class FileEncryptDecrypt {
    private static final int SHIFT = 3; // encryption key
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("FILE ENCRYPTION / DECRYPTION");
        System.out.println("1. Encrypt File");
        System.out.println("2. Decrypt File");
        System.out.print("Choose an option (1 or 2): ");
        int choice = sc.nextInt();
        sc.nextLine(); // consume newline
        System.out.print("Enter input file path: ");
        String inputFile = sc.nextLine();
        System.out.print("Enter output file path: ");
        String outputFile = sc.nextLine();
        try {
            if (choice == 1) {
                processFile(inputFile, outputFile, SHIFT);
                System.out.println("File encrypted successfully.");
            } else if (choice == 2) {
                processFile(inputFile, outputFile, -SHIFT);
                System.out.println("File decrypted successfully.");
            } else {
                System.out.println("Invalid choice.");
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
    static void processFile(String inputFile, String outputFile, int shift) {
        try (BufferedReader reader = new BufferedReader(new FileReader(inputFile));
             BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String encryptedLine = "";
                for (char c : line.toCharArray()) {
                    if (Character.isLetter(c)) {
                        char shiftedChar = (char) ((c - 'A' + shift) % 26 + 'A');
                        encryptedLine += shiftedChar;
                    } else {
                        encryptedLine += c;
                    }
                }
                writer.write(encryptedLine);
                writer.newLine();
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

The screenshot shows a Java development environment interface. The code editor displays a file named `FileEncryptDecrypt.java` with the following content:

```
FileEncryptDecrypt.java
private static void processFile(String input, String output, int shift) throws IOException {
    BufferedReader reader = new BufferedReader(new FileReader(input));
    BufferedWriter writer = new BufferedWriter(new FileWriter(output));
    int ch;
    while ((ch = reader.read()) != -1) {
        writer.write(ch + shift);
    }
    reader.close();
    writer.close();
}
```

The terminal window shows the following command-line interaction:

```
PS C:\Users\Aaryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks> cd "c:\Users\Aaryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks\" ; if ($?) { javac FileEncryptDecrypt.java } ; if ($?) { java FileEncryptDecrypt }
Error: Could not find or load main class FileEncryptDecrypt
Caused by: java.lang.ClassNotFoundException: FileEncryptDecrypt
PS C:\Users\Aaryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks> cd "c:\Users\Aaryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks\" ; if ($?) { javac FileEncryptDecrypt.java } ; if ($?) { java FileEncryptDecrypt }
== FILE ENCRYPTION / DECRYPTION ==
1. Encrypt File
2. Decrypt File
Choose (1 or 2): 1
Enter input file path: Desktop
Enter output file path: Tasks
Error: Desktop (The system cannot find the file specified)
PS C:\Users\Aaryan\Desktop\Cognifyz Internship Tasks\Level 2 Tasks>
```

The right sidebar includes a "Build with Agent" section and a "SUGGESTED ACTIONS" panel with options like "Build Workspace" and "Show Config". A watermark for "Activate Windows" is visible in the bottom right corner.