NAME:GANESH N MOROLIYA

Level 2
Task: Tic-Tac-Toe Game Task 1

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.

=>

```
import java.util.*;
public class Task2 {
    private static char[][] board = new char[3][3];
    private static char currentPlayer = 'X';
    private static boolean gameOn = true;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String playAgain;
        do {
            resetBoard();
            gameOn = true;
            currentPlayer = 'X';
            while (gameOn) {
                printBoard();
                playerMove(scanner);
                checkGameState();
                currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
            }
            System.out.print("Do you want to play again? (yes/no): ");
```

```
            playAgain = scanner.nextLine().trim().toLowerCase();
        } while (playAgain.equals("yes"));
    }

    private static void resetBoard() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                board[i][j] = '-';
            }
        }
    }

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

    private static void playerMove(Scanner scanner) {
        int row, col;
        while (true) {
            System.out.println("Player " + currentPlayer + ", enter your move (row and
column: 0, 1, or 2): ");
            row = scanner.nextInt();
            col = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            if (row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == '-') {
```

```
                board[row][col] = currentPlayer;
                break;
            } else {
                System.out.println("This move is not valid");
            }
        }
    }

    private static void checkGameState() {
        if (checkWinner()) {
            printBoard();
            System.out.println("Player " + currentPlayer + " wins!");
            gameOn = false;
        } else if (checkDraw()) {
            printBoard();
            System.out.println("The game is a draw!");
            gameOn = false;
        }
    }

    private static boolean checkWinner() {
        // Check rows and columns
        for (int i = 0; i < 3; i++) {
            if (board[i][0] == currentPlayer && board[i][1] == currentPlayer && board[i][2]
== currentPlayer) {
                return true;
            }
            if (board[0][i] == currentPlayer && board[1][i] == currentPlayer && board[2][i]
== currentPlayer) {
                return true;
            }
```

```
        }
        // Check diagonals
        if (board[0][0] == currentPlayer && board[1][1] == currentPlayer && board[2][2] == currentPlayer) {
            return true;
        }
        if (board[0][2] == currentPlayer && board[1][1] == currentPlayer && board[2][0] == currentPlayer) {
            return true;
        }
        return false;
    }

    private static boolean checkDraw() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board[i][j] == '-') {
                    return false;
                }
            }
        }
        return true;
    }
}
```

output:

Current board:

- - -

- - -

- - -

Player X, enter your move (row and column: 0, 1, or 2):

0

1

Current board:

- X -

- - -

- - -

Player O, enter your move (row and column: 0, 1, or 2):

2

1

Current board:

- X -

- - -

- O -

Player X, enter your move (row and column: 0, 1, or 2):

0

2

Current board:

- X X

- - -

- O -

Player O, enter your move (row and column: 0, 1, or 2):

0

0

Current board:

O X X

- - -

- O -

Player X, enter your move (row and column: 0, 1, or 2):

1

2

Current board:

O X X

- - X

- O -

Player O, enter your move (row and column: 0, 1, or 2):

3

0

This move is not valid

Player O, enter your move (row and column: 0, 1, or 2):

2

0

Current board:

O X X

- - X

O O -

Player X, enter your move (row and column: 0, 1, or 2):

2

2

Current board:

O X X

- - X

O O X

Player X wins!

Do you want to play again? (yes/**no):**

Task: Password Strength Checker Task 2

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.

=>

```
import java.util.*;

public class Task3 {


    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter your password: ");
        String password = scanner.nextLine();


        String strength = checkPasswordStrength(password);
        System.out.println("Password strength: " + strength);


    }


    public static String checkPasswordStrength(String password) {
        int length = password.length();
        boolean hasUpper = false;
        boolean hasLower = false;
        boolean hasDigit = false;
        boolean hasSpecial = false;


        for (char c : password.toCharArray()) {
            if (Character.isUpperCase(c)) {
```

```
                hasUpper = true;
            } else if (Character.isLowerCase(c)) {
                hasLower = true;
            } else if (Character.isDigit(c)) {
                hasDigit = true;
            } else if (isSpecialCharacter(c)) {
                hasSpecial = true;
            }
        }

        if (length >= 12 && hasUpper && hasLower && hasDigit && hasSpecial) {
            return "Very Strong";
        } else if (length >= 8 && hasUpper && hasLower && hasDigit) {
            return "Strong";
        } else if (length >= 8 && (hasUpper || hasLower) && hasDigit) {
            return "Medium";
        } else if (length >= 6) {
            return "Weak";
        } else {
            return "Very Weak";
        }
    }

    public static boolean isSpecialCharacter(char c) {
        String specialCharacters = "!@#$%^&*()-+";
        return specialCharacters.indexOf(c) != -1;
    }
}
```

Output:

1) Enter your password: RCOem@1234#

Password strength: Strong

2) Enter your password: rhsk12

Password strength: Weak

Task: File Encryption/Decryption Task 3

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.

=>

```
package Level2;

import java.io.*;

import java.util.Scanner;


public class Task4{


    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        System.out.println("Do you want to (E)ncrypt or (D)ecrypt a file?");

        String choice = scanner.nextLine().trim().toUpperCase();


        if (!choice.equals("E") && !choice.equals("D")) {

            System.out.println("Invalid choice. Please enter E for Encryption or D for
Decryption.");

            return;

        }


        System.out.println("Enter the file path: ");

        String filePath = scanner.nextLine().trim();


        System.out.println("Enter the shift key (integer): ");
```

```
    int shiftKey = scanner.nextInt();

    if (choice.equals("D")) {
        shiftKey = -shiftKey;
    }

    try {
        String content = readFile(filePath);
        String processedContent = processContent(content, shiftKey);
        String outputFilePath = choice.equals("E") ? "encrypted.txt" : "decrypted.txt";
        writeFile(outputFilePath, processedContent);
        System.out.println("The file has been processed. Output file: " + outputFilePath);
    } catch (IOException e) {
        System.out.println("Error processing the file: " + e.getMessage());
    }
}

private static String readFile(String filePath) throws IOException {
    StringBuilder content = new StringBuilder();
    try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
        String line;
        while ((line = br.readLine()) != null) {
            content.append(line).append(System.lineSeparator());
        }
    }
    return content.toString();
}

private static void writeFile(String filePath, String content) throws IOException {
    try (BufferedWriter bw = new BufferedWriter(new FileWriter(filePath))) {
```

```
            bw.write(content);
        }
    }

    private static String processContent(String content, int shiftKey) {
        StringBuilder result = new StringBuilder();
        for (char c : content.toCharArray()) {
            if (Character.isLetter(c)) {
                char base = Character.isUpperCase(c) ? 'A' : 'a';
                char shifted = (char) (((c - base + shiftKey) % 26 + 26) % 26 + base);
                result.append(shifted);
            } else {
                result.append(c);
            }
        }
        return result.toString();
    }
}
```

Output:

```
Do you want to (E)ncrypt or (D)ecrypt a file?
E
Enter the file path:
GANESHMOROLIYA.txt
Enter the shift key (integer):
5
Error processing the file: GANESHMOROLIYA.txt (The system cannot find the file specified)
```