MODULE 2:

1.Java Foundations Final Project

Overview

The final project gives you an opportunity to showcase your skills with the programming concepts presented in the Java Foundations course. Develop an application, animation, game, or anything else that demonstrates your knowledge of foundational Java programming concepts.

Tasks

Develop an application, animation, game, or anything else that demonstrates your knowledge of foundational Java programming concepts. You may find it helpful to review the Spiral Model of development and design a plan before you begin programming.

Your program must include the following:

- **At least 2 classes, not including the test class**
  - **Encapsulate these, and additional classes**
  - **Provide at least 2 fields and 2 methods in every class, not including getters or setters.**
- **An Array or ArrayList**
- **The ability to accept user-input at least once. Always accept user input without crashing.**

You're welcome to base your program on any of the following examples or create your own scenario:

- **Model an employee database. Supply a collection of employees. Provide methods to order or display employees by their salary and productivity ratings. Allow the manager to fire all employees who earn above-average salaries but offer below-average productivity. Allow the manager to hire new employees.**
- **Develop a meal-planner which assembles a list of food a person can eat in one day based on user-specified calorie limit and an even larger list of every possible food included in the program.**
- **Create a text choose-your-own-adventure game**
- **Create a Tic-Tac-Toe game (Noughts and Crosses)**

You may use previous problem sets, exercises, or lecture notes as inspiration to help you get started, but your program must be significantly different from these materials. For example, you're not allowed to simply change Problem Set 8's soccer league to a

cricket league.
Include a document with a brief description of software.

Project Overview: Tic-Tac-Toe Game

Objective: Develop a Tic-Tac-Toe game that demonstrates foundational Java programming concepts, including class encapsulation, user input, and array manipulation. The game will allow two players to play against each other on the console.

Project Structure

Classes:

1. **Game**
2. **Player**

Additional Requirements:

- Use of arrays to manage the game board.
- Accept user input to make moves.
- Ensure input is validated to avoid crashes.
- Implement methods to check game status (win, draw, ongoing).

Class Descriptions

1. Game Class

This class manages the game board, player turns, and game status.

Fields:

- char[][] board - 2D array representing the Tic-Tac-Toe board.
- Player[] players - Array holding the two players.

Methods:

- initializeBoard() - Initializes or resets the game board.
- printBoard() - Displays the current state of the board.

- makeMove(int row, int col) - Allows a player to make a move.
- checkWin() - Checks if there is a winner.
- checkDraw() - Checks if the game is a draw.
- playGame() - Main game loop for handling player turns and game status.

2. Player Class

This class represents a player in the game.

Fields:

- String name - The player's name.
- char symbol - The player's symbol ('X' or 'O').

Methods:

- getName() - Returns the player's name.
- getSymbol() - Returns the player's symbol.

Implementation

Here's a simplified version of how you might implement these classes:

Player Class

```
public class Player {
    private String name;
    private char symbol;

    public Player(String name, char symbol) {
        this.name = name;
        this.symbol = symbol;
    }

    public String getName() {
        return name;
```

```
    }

    public char getSymbol() {
        return symbol;
    }
}
```

Game Class

```
import java.util.Scanner;

public class Game {
    private char[][] board;
    private Player[] players;
    private int currentPlayerIndex;

    public Game(Player player1, Player player2) {
        board = new char[3][3];
        players = new Player[]{player1, player2};
        currentPlayerIndex = 0;
        initializeBoard();
    }

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

```
    }
}

public void printBoard() {
    System.out.println(" " + board[0][0] + " | " + board[0][1] + " | " + board[0][2]);
    System.out.println("-----------");
    System.out.println(" " + board[1][0] + " | " + board[1][1] + " | " + board[1][2]);
    System.out.println("-----------");
    System.out.println(" " + board[2][0] + " | " + board[2][1] + " | " + board[2][2]);
}

public boolean makeMove(int row, int col) {
    if (row < 0 || row >= 3 || col < 0 || col >= 3 || board[row][col] != ' ') {
        return false;
    }
    board[row][col] = players[currentPlayerIndex].getSymbol();
    return true;
}

public boolean checkWin() {
    char symbol = players[currentPlayerIndex].getSymbol();

    // Check rows, columns, and diagonals
    for (int i = 0; i < 3; i++) {
        if ((board[i][0] == symbol && board[i][1] == symbol && board[i][2] == symbol) ||
            (board[0][i] == symbol && board[1][i] == symbol && board[2][i] == symbol)) {
```

```
            return true;
        }
    }
    if ((board[0][0] == symbol && board[1][1] == symbol && board[2][2] == symbol) ||
        (board[0][2] == symbol && board[1][1] == symbol && board[2][0] == symbol)) {
        return true;
    }
    return false;
}

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

public void playGame() {
    Scanner scanner = new Scanner(System.in);
    while (true) {
        printBoard();
        Player currentPlayer = players[currentPlayerIndex];
```

```
        System.out.println(currentPlayer.getName() + "'s turn (" + currentPlayer.getSymbol() +
")");
        int row, col;
        while (true) {
            System.out.print("Enter row (0-2): ");
            row = scanner.nextInt();
            System.out.print("Enter column (0-2): ");
            col = scanner.nextInt();
            if (makeMove(row, col)) {
                break;
            } else {
                System.out.println("Invalid move. Try again.");
            }
        }
        if (checkWin()) {
            printBoard();
            System.out.println(currentPlayer.getName() + " wins!");
            break;
        }
        if (checkDraw()) {
            printBoard();
            System.out.println("The game is a draw!");
            break;
        }
        currentPlayerIndex = 1 - currentPlayerIndex; // Switch player
    }
```

```
        scanner.close();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter name for Player 1: ");
        String name1 = scanner.nextLine();
        System.out.print("Enter name for Player 2: ");
        String name2 = scanner.nextLine();

        Player player1 = new Player(name1, 'X');
        Player player2 = new Player(name2, 'O');

        Game game = new Game(player1, player2);
        game.playGame();
    }
}
```