Filipe Lutz Mariano - 25956

**Mobile Apps 2**

**Assignment one.**

**Submission Date 11/10/2024**

Percent 10%

Include all workings and references

## *Section A.*

1. Explain the main advantages and disadvantages of using the MVC design pattern (3 marks).

   The MVC pattern divides an application into three interconnected components to separate internal data (Model), user interface (View), and control logic (Controller).

   Advantages:

   - Separation of Concerns: MVC promotes a clear division of responsibilities. The model handles the data and business logic, the view manages the UI, and the controller acts as the intermediary. This leads to cleaner and more maintainable code.
   - Reusability: Components in MVC can be reused. For instance, the same model can be used for different views or applications.
   - Scalability: The separation of logic makes it easier to scale applications since each component can be developed and modified independently without affecting the others.
   - Parallel Development: Different developers can work on different components (model, view, controller) in parallel, which speeds up development.

   Disadvantages:

   - Complexity in Smaller Applications: For simple applications, MVC can add unnecessary complexity, as dividing into three components can lead to over-engineering.
   - Increased Learning Curve: Developers need to understand the separation of concerns and the interactions between MVC components, which might require a steeper learning curve, especially for beginners.

- Maintenance Challenges: While MVC promotes separation, in larger projects, tight coupling may develop over time if care is not taken, leading to difficulties in maintenance.

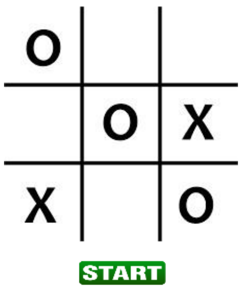
2. Explain what is meant by dependency injection and how does this concept aid MVC (2 marks).

Dependency Injection is a design pattern where an object receives other objects, rather than creating those dependencies internally. In simple terms, instead of the class instantiating its dependencies, they are "injected" into it, usually via the constructor, methods, or properties.

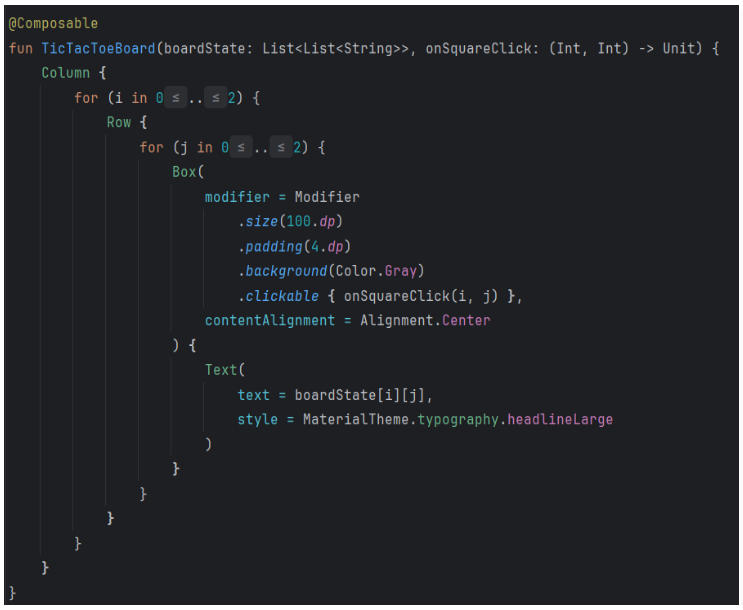
Dependency Injection aids MVC:

- Loose Coupling: Dependency Injection decouples classes in an MVC architecture. The controller, model, or view can depend on abstractions (like interfaces) instead of concrete implementations. This makes the code more modular and easier to modify.
- Testability: It makes unit testing easier because dependencies can be mocked or stubbed, allowing each component (especially controllers) to be tested independently.
- Flexibility: DI allows components to be swapped easily. For example, if a service layer needs to be replaced with a new implementation, it can be injected without changing the controller logic.

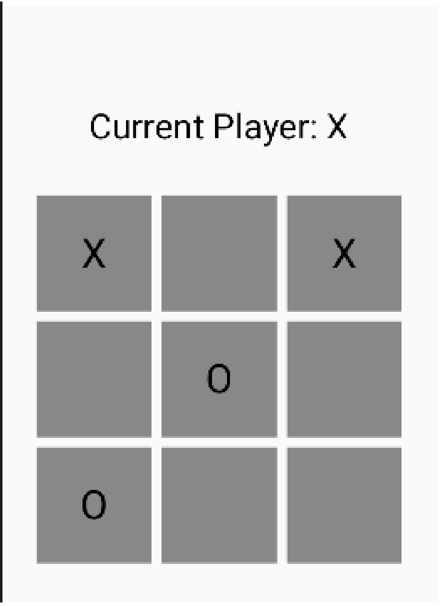
***Section B.***



1. Set up the Compose graphical settings for an X's and O's game within an Android project (2.5 marks)



```kotlin
@Composable
fun TicTacToeBoard(boardState: List<List<String>>, onSquareClick: (Int, Int) -> Unit) {
    Column {
        for (i in 0 ≤.. ≤ 2) {
            Row {
                for (j in 0 ≤.. ≤ 2) {
                    Box(
                        modifier = Modifier
                            .size(100.dp)
                            .padding(4.dp)
                            .background(Color.Gray)
                            .clickable { onSquareClick(i, j) },
                        contentAlignment = Alignment.Center
                    ) {
                        Text(
                            text = boardState[i][j],
                            style = MaterialTheme.typography.headlineLarge
                        )
                    }
                }
            }
        }
    }
}
```



2. Whilst not including the logic of the code, explain how you would

   . Show how a specific square was selected and could not be re-selected.

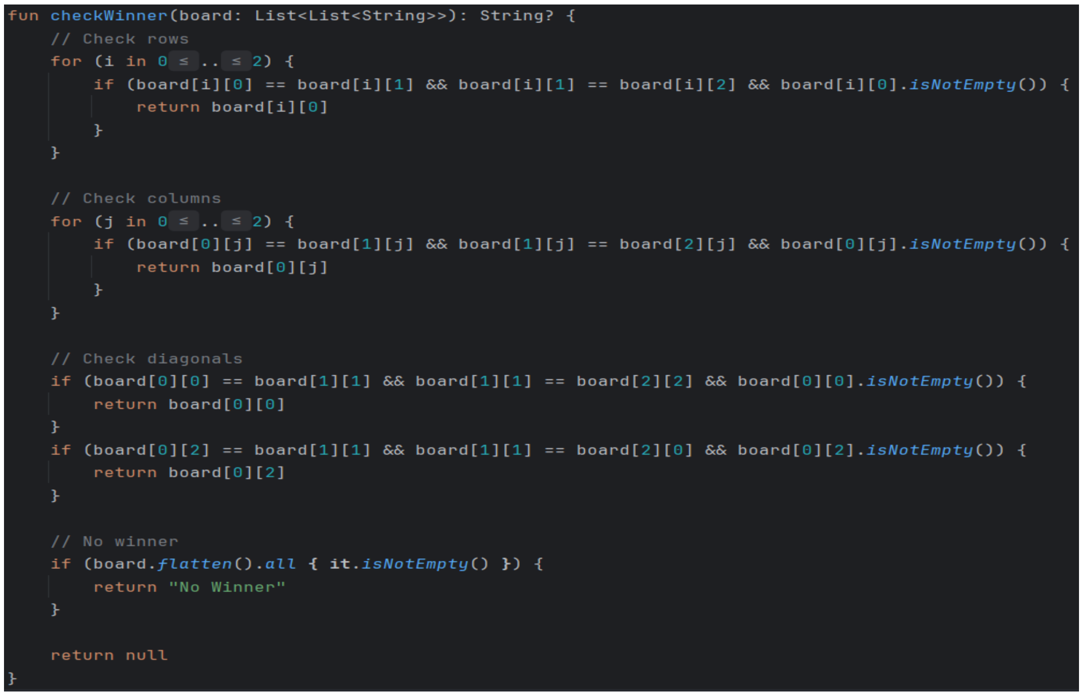
To ensure that a square cannot be re-selected once clicked, I'd need to maintain the state of each cell. I used a state management solution in Jetpack Compose. For example, I used a **{** *mutableStateOf*(*List*(3) **{** *MutableList*(3) **{** "" **}** **}**) **}**
to store the state of each cell.

   . Have a general algorithm to calculate if someone has won the game.

**Pseudocode:**

1. **Check rows:** If all the cells in any row contain the same non-empty value, declare a win.
2. **Check columns:** If all the cells in any column contain the same non-empty value, declare a win.
3. **Check diagonals:** There are two diagonals to check. If all cells in any diagonal contain the same non-empty value, declare a win.
4. If no rows, columns, or diagonals meet the win condition and all cells are filled, declare a draw.

Include any settings within the project itself, but you can write the algorithm in a Word or text document. Include within zip drive of your project.



```
fun checkWinner(board: List<List<String>>): String? {
    // Check rows
    for (i in 0 ≤ .. ≤ 2) {
        if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i][0].isNotEmpty()) {
            return board[i][0]
        }
    }

    // Check columns
    for (j in 0 ≤ .. ≤ 2) {
        if (board[0][j] == board[1][j] && board[1][j] == board[2][j] && board[0][j].isNotEmpty()) {
            return board[0][j]
        }
    }

    // Check diagonals
    if (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0].isNotEmpty()) {
        return board[0][0]
    }
    if (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2].isNotEmpty()) {
        return board[0][2]
    }

    // No winner
    if (board.flatten().all { it.isNotEmpty() }) {
        return "No Winner"
    }

    return null
}
```

## References:

1. MVC Design Pattern
   https://www.geeksforgeeks.org/mvc-design-pattern/

2. Dependency injection in Android
   https://developer.android.com/training/dependency-injection

3. Understanding Dependency Injection
   https://www.telerik.com/blogs/aspnet-core-basics-understanding-dependency-injection

4. JetBrains, "State in Jetpack Compose,"
   https://developer.android.com/develop/ui/compose/state

5. Tic Tac Toe Game in Jetpack Compose
   https://www.youtube.com/watch?v=Q-3xkyih2uQ

6. https://www.youtube.com/watch?v=KuNWkaoaWjc

GitHub Repository TicTacToe:

- https://github.com/FilipeLutz/TicTacToe