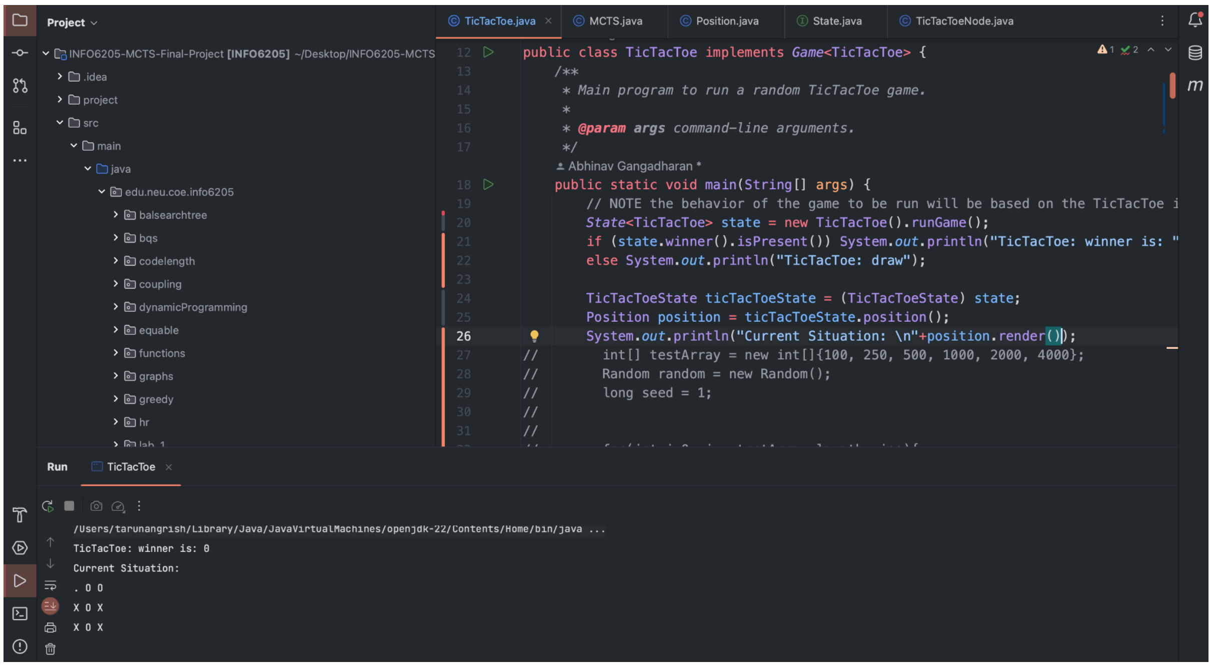
# INFO6205 Program Structures and Algorithms
# Spring 2024

Abhinav Gangadharan - NUID - 002889956
Tarun Angrish - NUID - 002807094
Github - https://github.com/tarunangrish-neu/INFO6205-MCTS-Final-Project

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

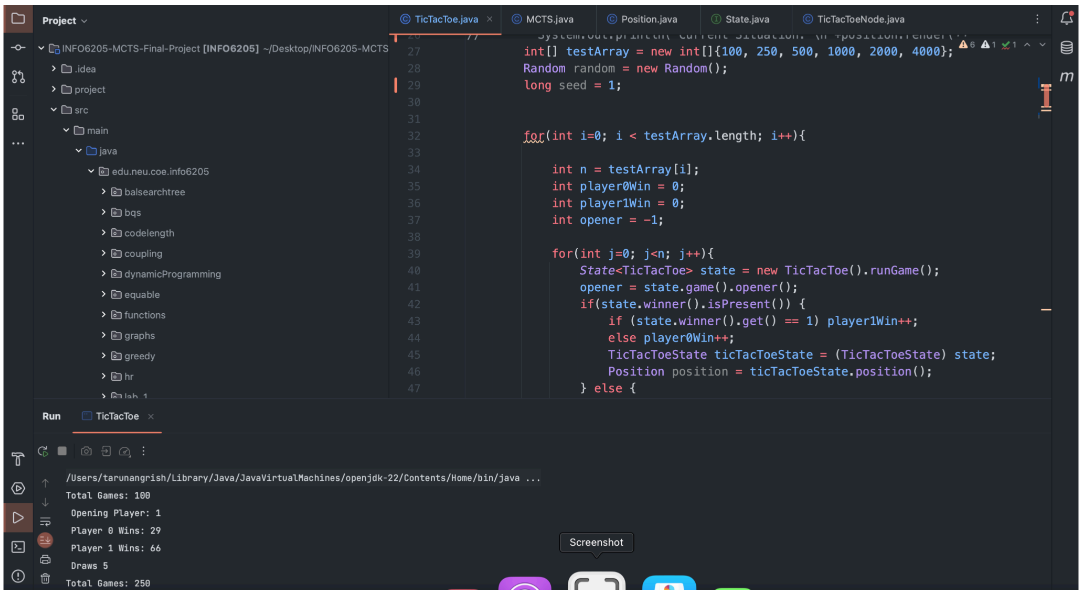


```java
public class TicTacToe implements Game<TicTacToe> {
    /**
     * Main program to run a random TicTacToe game.
     *
     * @param args command-line arguments.
     */
    public static void main(String[] args) {
        // NOTE the behavior of the game to be run will be based on the TicTacToe i
        State<TicTacToe> state = new TicTacToe().runGame();
        if (state.winner().isPresent()) System.out.println("TicTacToe: winner is: "
        else System.out.println("TicTacToe: draw");

        TicTacToeState ticTacToeState = (TicTacToeState) state;
        Position position = ticTacToeState.position();
        System.out.println("Current Situation: \n"+position.render());
//        int[] testArray = new int[]{100, 250, 500, 1000, 2000, 4000};
//        Random random = new Random();
//        long seed = 1;
//
//
```

```
/Users/tarunangrish/Library/Java/JavaVirtualMachines/openjdk-22/Contents/Home/bin/java ...
TicTacToe: winner is: 0
Current Situation:
. 0 0
X 0 X
X 0 X
```

Fig. 1 Implementation of Tic-Tac-Toe

Experiment 1: Running the game on the default setting



```java
        int[] testArray = new int[]{100, 250, 500, 1000, 2000, 4000};
        Random random = new Random();
        long seed = 1;


        for(int i=0; i < testArray.length; i++){

            int n = testArray[i];
            int player0Win = 0;
            int player1Win = 0;
            int opener = -1;

            for(int j=0; j<n; j++){
                State<TicTacToe> state = new TicTacToe().runGame();
                opener = state.game().opener();
                if(state.winner().isPresent()) {
                    if (state.winner().get() == 1) player1Win++;
                    else player0Win++;
                    TicTacToeState ticTacToeState = (TicTacToeState) state;
                    Position position = ticTacToeState.position();
                } else {
```

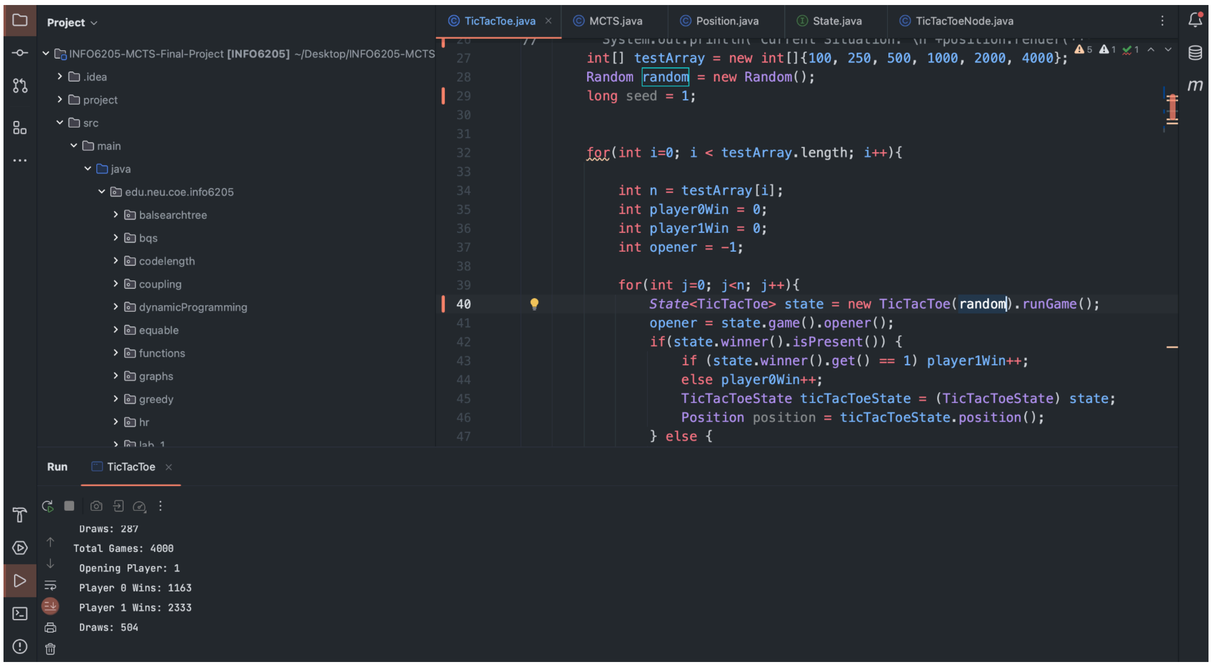
```
/Users/tarunangrish/Library/Java/JavaVirtualMachines/openjdk-22/Contents/Home/bin/java ...
Total Games: 100
 Opening Player: 1
 Player 0 Wins: 29
 Player 1 Wins: 66
 Draws 5
Total Games: 250
```

Fig 2. Implementation of Experimental Framework to test run with default setting(s)

**Output with the default setting [No seed]:**

```
Total Games: 100
 Opening Player: 1
 Player 0 Wins: 24
 Player 1 Wins: 70
 Draws: 6

Total Games: 250
 Opening Player: 1
 Player 0 Wins: 97
 Player 1 Wins: 127
 Draws: 26

Total Games: 500
 Opening Player: 1
 Player 0 Wins: 254
 Player 1 Wins: 191
 Draws: 55

Total Games: 1000
 Opening Player: 1
 Player 0 Wins: 249
 Player 1 Wins: 597
 Draws: 154

Total Games: 2000
 Opening Player: 1
 Player 0 Wins: 412
 Player 1 Wins: 1471
 Draws: 117

Total Games: 4000
 Opening Player: 1
 Player 0 Wins: 784
 Player 1 Wins: 2600
 Draws: 616
```

Experiment 2: Running the game with a random seed.



```java
        int[] testArray = new int[]{100, 250, 500, 1000, 2000, 4000};
        Random random = new Random();
        long seed = 1;


        for(int i=0; i < testArray.length; i++){

            int n = testArray[i];
            int player0Win = 0;
            int player1Win = 0;
            int opener = -1;

            for(int j=0; j<n; j++){
                State<TicTacToe> state = new TicTacToe(random).runGame();
                opener = state.game().opener();
                if(state.winner().isPresent()) {
                    if (state.winner().get() == 1) player1Win++;
                    else player0Win++;
                    TicTacToeState ticTacToeState = (TicTacToeState) state;
                    Position position = ticTacToeState.position();
                } else {
```

```
 Draws: 287
Total Games: 4000
 Opening Player: 1
 Player 0 Wins: 1163
 Player 1 Wins: 2333
 Draws: 504
```

Fig 3. Implementation of Experimental Framework to test run with random seed

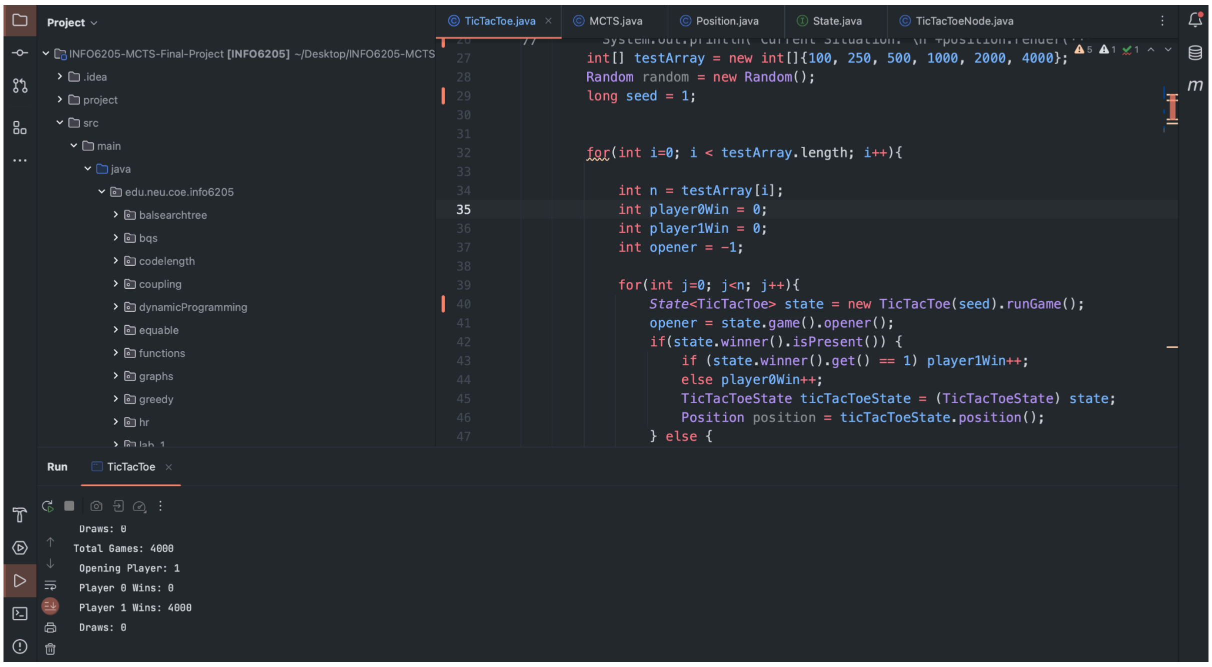
**Output:**

```
Total Games: 100
 Opening Player: 1
 Player 0 Wins: 38
 Player 1 Wins: 47
 Draws: 15

Total Games: 250
 Opening Player: 1
 Player 0 Wins: 67
 Player 1 Wins: 157
 Draws: 26

Total Games: 500
 Opening Player: 1
 Player 0 Wins: 140
 Player 1 Wins: 303
 Draws: 57

Total Games: 1000
 Opening Player: 1
 Player 0 Wins: 326
 Player 1 Wins: 549
 Draws: 125

Total Games: 2000
 Opening Player: 1
 Player 0 Wins: 558
 Player 1 Wins: 1155
```

Draws: 287

Total Games: 4000
Opening Player: 1
Player 0 Wins: 1163
Player 1 Wins: 2333
Draws: 504

Experiment 3: Running the game with seed = 1



```java
        int[] testArray = new int[]{100, 250, 500, 1000, 2000, 4000};
        Random random = new Random();
        long seed = 1;


        for(int i=0; i < testArray.length; i++){

            int n = testArray[i];
            int player0Win = 0;
            int player1Win = 0;
            int opener = -1;

            for(int j=0; j<n; j++){
                State<TicTacToe> state = new TicTacToe(seed).runGame();
                opener = state.game().opener();
                if(state.winner().isPresent()) {
                    if (state.winner().get() == 1) player1Win++;
                    else player0Win++;
                    TicTacToeState ticTacToeState = (TicTacToeState) state;
                    Position position = ticTacToeState.position();
                } else {
```

```
 Draws: 0
Total Games: 4000
 Opening Player: 1
 Player 0 Wins: 0
 Player 1 Wins: 4000
 Draws: 0
```

Fig 4. Implementation of Experimental Framework to test run with seed=1

Total Games: 100
Opening Player: 1
Player 0 Wins: 0
Player 1 Wins: 100
Draws: 0

Total Games: 250
Opening Player: 1
Player 0 Wins: 0
Player 1 Wins: 250
Draws: 0

Total Games: 500
Opening Player: 1
Player 0 Wins: 0
Player 1 Wins: 500
Draws: 0

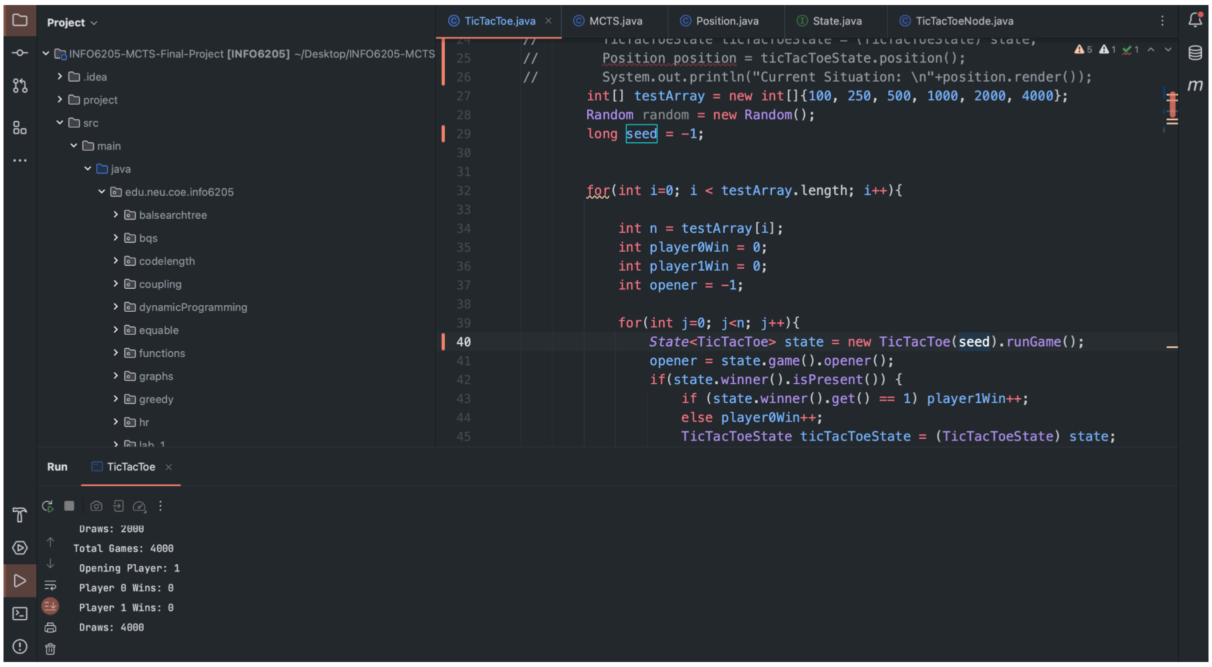
Total Games: 1000
Opening Player: 1

Player 0 Wins: 0
Player 1 Wins: 1000
Draws: 0

Total Games: 2000
Opening Player: 1
Player 0 Wins: 0
Player 1 Wins: 2000
Draws: 0

Total Games: 4000
Opening Player: 1
Player 0 Wins: 0
Player 1 Wins: 4000
Draws: 0

Experiment 4: Running the game with a seed of -1



```java
//        Position position = ticTacToeState.position();
//        System.out.println("Current Situation: \n"+position.render());
        int[] testArray = new int[]{100, 250, 500, 1000, 2000, 4000};
        Random random = new Random();
        long seed = -1;


        for(int i=0; i < testArray.length; i++){

            int n = testArray[i];
            int player0Win = 0;
            int player1Win = 0;
            int opener = -1;

            for(int j=0; j<n; j++){
                State<TicTacToe> state = new TicTacToe(seed).runGame();
                opener = state.game().opener();
                if(state.winner().isPresent()) {
                    if (state.winner().get() == 1) player1Win++;
                    else player0Win++;
                    TicTacToeState ticTacToeState = (TicTacToeState) state;
```

```
  Draws: 2000
Total Games: 4000
  Opening Player: 1
  Player 0 Wins: 0
  Player 1 Wins: 0
  Draws: 4000
```

Fig 5. Implementation of Experimental Framework to test run with seed = -1

Total Games: 100
Opening Player: 1
Player 0 Wins: 0
Player 1 Wins: 0
Draws: 100

Total Games: 250
Opening Player: 1
Player 0 Wins: 0
Player 1 Wins: 0
Draws: 250

Total Games: 500
Opening Player: 1

```
 Player 0 Wins: 0
 Player 1 Wins: 0
 Draws: 500

Total Games: 1000
 Opening Player: 1
 Player 0 Wins: 0
 Player 1 Wins: 0
 Draws: 1000

Total Games: 2000
 Opening Player: 1
 Player 0 Wins: 0
 Player 1 Wins: 0
 Draws: 2000

Total Games: 4000
 Opening Player: 1
 Player 0 Wins: 0
 Player 1 Wins: 0
 Draws: 4000
```

**Conclusion**:
From Figure 1, we can see that our code has the correct implementation for the Tic-Tac-Toe game. Our subsequent tests can also prove that our game implementation is in line with the actual situation that happens while playing the game. We can observe from the experiments that the relationship between the two players is not one-sided, which is true in the case of random generation. At the same time, the winning rate is always more favorable for the person who gets to have the more number of moves. In tic-tac-toe, a game can have only nine moves at most, one player can take five, and another player takes four steps. So, the player who takes five steps will have a particular advantage, also reflected in the final winning rate. Additionally, it was also observed that using seeds -1 and 1 creates deterministic conditions that always result in a win for the player who goes first. On the other hand, a seed of 0 results in all draws, indicating an implementation that ensures no victories.