Principles of Programming: Coursework 2 – Requirement 4

Connect N: Report discussing code restructuring to reflect fundamental object-oriented program concepts.

Submitted by: Shayaan Khatri

Username: ssk57

After the game Connect4 has been implemented in Java using object-oriented programming principles, a few steps can be taken to convert the game from Connect4 to ConnectN and that is what I did. Due to the nature of the object-oriented programming techniques this task can be achieved in a relatively systematic way than if object-oriented programming was not used. The first step I took was that I created a new class called "ConnectN.Java" , which was exactly as my "Connect4.Java" class but with an ability to ask the user for input with regards to the game winning condition they want. I achieved this as I set integer gameCondition as a static so I can access it in the Board.java class without calling the ConnectN class. I also set the getter and setter to help me achieve this. I was able to access the game condition by using the method getGameCondition(). I set the getGameCondition equals to the count of tokens matched required when checking for a win. This is how I instilled the condition while checking for a win. This is demonstrated in the screenshots below. I will get my N conditions for the game with this.

For the 3rd player: in the playGame method() of my ConnectN class. I was able to add a third player which will be called bot2. This player is added to the array list that is named "players" so now I have a total of 3 players. I then got the size of this array list by doing players.size() and by applying the remainder technique, I was able to that ensure all three players can play in the game.

```java
import java.io.InputStreamReader;
import java.util.ArrayList;


//Class to set the working logic of the Connect 4 game
public class ConnectN {
    private Board board;
    public static int gameCondition;


    public ConnectN() {
        board = new Board();
        setGameCondition();
    }

    public static void setGameCondition(){
        System.out.println("Please input an N Condition between 2 < N < 7:");
        BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
        try {
            gameCondition = Integer.parseInt(input.readLine());
        } catch (IOException e) {
            System.out.println("Please only input relevant integer");;
        }

    }

    //Getter function to get the game condition
    public static int getGameCondition() {
        return gameCondition;
    }
```

```java
public void playGame(){
    //Method called to print the game instructions
    gameInstructions();
    //The board is printed
    System.out.println(board);
    //Makes the user aware that an input is required
    System.out.println("Please input a token in any COLUMN 1 to 7 below:");

    //Creates a human player and a bot player using the HumanPlayer and ComputerPlayer class.
    HumanPlayer human = new HumanPlayer(new Token('R'));
    ComputerPlayer bot1 = new ComputerPlayer(new Token('Y'));
    ComputerPlayer bot2 = new ComputerPlayer(new Token('B'));
    //An ArrayList is created and called players
    ArrayList<Player> players = new ArrayList<>();
    //The players are added to this ArrayList
    players.add(human);
    players.add(bot1);
    players.add(bot2);


    //currentPlayer is set as an integer type which initiates as zero
    int currentPlayer = 0;

    while (true) {
        //The total player size for connectN is 2 and by using the remainder function
        //When it human players turn, the currentPlayer variable is set to 0
        //When it is the bot player turn then the variable increases to 1
        //The remainder function is used so values only stay between 0 and 1.
        //So human and bot keep changing turns this way
        Player current = players.get(currentPlayer % players.size());
```

```java
    //method to check for horizontal tokens pattern
    private boolean checkHorizontal(Token player) {
        //counter is initialized at 0
        int count = 0;
        //iteration over the rows
        for (int row = 0; row < board.length; row++) {
            //iteration over the columns
            for (int column = 0; column < board[row].length; column++) {
                if (board[row][column] == player) {
                    count = count + 1;
                    if (count >= ConnectN.getGameCondition()) {
                        return true;
                    }
                } else {
                    //coun
                    count = 0;
                }
            }
            count = 0;
        }
        return false;
    }
```