```
import java.awt.Color;
import info.gridworld.grid.*;

public class ConnectFourGame
{
    private int[][] board;
    private int currentTurnNumber;

    public ConnectFourGame(int rows, int cols)
    {
        board = new int[rows][cols];
        // you need to fill board with 0's
        for (int r=0; r<rows; r++)
            for (int c=0; c<cols; c++)
                board[r][c] = 0;

        currentTurnNumber = 1;
    }

    // returns 0, 1, or 2
    public int getWinner()
    {
        //check horizontal
        for(int r=0; r<board.length; r++){
            //check black
            int counter = 0;
            for(int c=0; c<board[0].length; c++){
                if(board[r][c] == 1) counter++;
                else counter = 0;

                if(counter == 4) return 1;

            }
            //check red
            counter = 0;
            for(int c=0; c<board[0].length; c++){

                if(board[r][c] == 2) counter++;
                else counter = 0;

                if(counter ==4) return 2;

            }
        }
        //check vertical
        for(int c=0; c<board[0].length; c++){
            //check black
            int counter = 0;
            for(int r=0; r<board.length; r++){
                if(board[r][c] == 1) counter++;
                else counter = 0;

                if(counter == 4) return 1;

            }
```

```
            //check red
            counter = 0;
            for(int r=0; r<board.length; r++){
                if(board[r][c] == 2) counter++;
                else counter = 0;
                if(counter ==4) return 2;
            }
        }
        //check dr diag
        for(int startCol=0; startCol<board[0].length; startCol++){
            for(int startRow=0; startRow<board.length; startRow++){
                int counter = 0;
                //check black
                for(int i=0; i<4; i++){
                    if(startRow +i >= board.length || startCol + i >=
board[0].length)break;
                    if(board[startRow + i][startCol + i] == 1) counter++;
                    if(counter == 4) return 1;
                }
                counter =0;
                //check red
                for(int i=0; i<4; i++){
                    if(startRow + i >= board.length || startCol + i >=board[0].length)
break;
                    if(board[startRow + i][startCol + i] == 2) counter++;
                    if(counter == 4) return 2;
                }
            }
        }
        //check dl diag
        for(int startCol=0; startCol<board[0].length; startCol++) {
            for(int startRow=0; startRow<board.length; startRow++) {
                int counter = 0;
                //check black
                for(int i=0; i<4; i++){
                    if(startRow +i >=board.length || startCol - i <0)break;
                    if(board[startRow + i][startCol - i] == 1) counter++;
                    if(counter == 4) return 1;
                }

                counter =0;
                //check red
                for(int i=0; i<4; i++){
                    if(startRow +i >=board.length || startCol - i <0)break;
                    if(board[startRow + i][startCol - i] == 2) counter++;
                    if(counter == 4) return 2;
                }

            }
        }
        return 0;
    }

    // updates the state of the game (board and currentTurnColor)
    // returns the row in which the checker would end up
```

```
    // returns -1 if the column col is completely full and no checker can be dropped
    public int dropChecker(int col)
    {
        for(int r=board.length-1; r>=0; r--){
            if(board[r][col] == 0) {
                board[r][col] = currentTurnNumber;
                if(currentTurnNumber ==1) currentTurnNumber = 2;
                else currentTurnNumber = 1;
                return r;
            }
        }
        return -1;
    }
}
```