```
import java.io.*;

/**
 *
 * @author James Spargo
 * This class controls the game play for the Max Connect-Four game.
 * To compile the program, use the following command from the maxConnectFour directory:
 * javac *.java
 *
 * the usage to run the program is as follows:
 * ( again, from the maxConnectFour directory )
 *
 *  -- for interactive mode:
 * java MaxConnectFour interactive [ input_file ] [ computer-next / human-next ] [ search
depth]
 *
 * -- for one move mode
 * java maxConnectFour.MaxConnectFour one-move [ input_file ] [ output_file ] [ search
depth]
 *
 * description of arguments:
 *  [ input_file ]
 *  -- the path and filename of the input file for the game
 *
 *  [ computer-next / human-next ]
 *  -- the entity to make the next move. either computer or human. can be abbreviated to
either C or H. This is only used in interactive mode
 *
 *  [ output_file ]
 *  -- the path and filename of the output file for the game.  this is only used in one-
move mode
 *
 *  [ search depth ]
 *  -- the depth of the minimax search algorithm
 *
 *
 */

public class maxconnect4
{
  public static void main(String[] args)
  {
    // check for the correct number of arguments
    if( args.length != 4 )
    {
      System.out.println("Four command-line arguments are needed:\n"
                         + "Usage: java [program name] interactive [input_file] [computer-
next / human-next] [depth]\n"
                         + " or:  java [program name] one-move [input_file] [output_file]
[depth]\n");

      exit_function( 0 );
     }

    // parse the input arguments
    String game_mode = args[0].toString();                              // the game mode
    String input = args[1].toString();                                  // the input game
file
    int depthLevel = Integer.parseInt( args[3] );               // the depth level of the
ai search

    // create and initialize the game board
    GameBoard currentGame = new GameBoard( input );
    // create the Ai Player
    AiPlayer calculon = new AiPlayer();

    //  variables to keep up with the game
    int playColumn = 99;                                //  the players choice of column
to play
    boolean playMade = false;                   //  set to true once a play has been made

    if( game_mode.equalsIgnoreCase( "interactive" ) )
    {
        System.out.println("interactive mode is currently not implemented\n");
        return;
    }

    else if( !game_mode.equalsIgnoreCase( "one-move" ) )
    {
      System.out.println( "\n" + game_mode + " is an unrecognized game mode \n try again.
\n" );
      return;
    }

    /////////////   one-move mode ///////////
    // get the output file name
    String output = args[2].toString();                        // the output game file

    System.out.print("\nMaxConnect-4 game\n");
    System.out.print("game state before move:\n");

    //print the current game board
    currentGame.printGameBoard();
    // print the current scores
    System.out.println( "Score: Player 1 = " + currentGame.getScore( 1 ) +
                        ", Player2 = " + currentGame.getScore( 2 ) + "\n " );

    // ****************** this chunk of code makes the computer play
    if( currentGame.getPieceCount() < 42 )
    {
        int current_player = currentGame.getCurrentTurn();
        // AI play - random play
        playColumn = calculon.findBestPlay( currentGame );

        // play the piece
        currentGame.playPiece( playColumn );

        // display the current game board
        System.out.println("move " + currentGame.getPieceCount()
                           + ": Player " + current_player
                           + ", column " + playColumn);
        System.out.print("game state after move:\n");
        currentGame.printGameBoard();

        // print the current scores
        System.out.println( "Score: Player 1 = " + currentGame.getScore( 1 ) +
                            ", Player2 = " + currentGame.getScore( 2 ) + "\n " );

        currentGame.printGameBoardToFile( output );
    }
    else
    {
        System.out.println("\nI can't play.\nThe Board is Full\n\nGame Over");
    }

    //************************** end computer play
    return;

} // end of main()

  /**
   * This method is used when to exit the program prematurly.
   * @param value an integer that is returned to the system when the program exits.
   */
  private static void exit_function( int value )
  {
      System.out.println("exiting from MaxConnectFour.java!\n\n");
      System.exit( value );
  }
} // end of class connectFour
```