********** OUTPUT **********

It is failing almost instantly in trying to access the board… I have commented certain areas of code also (which are highlighted in red), since it causes the execution to fail altogether of the software. I am unsure of the cause…

Validation is ok for chip insertion

```
Welcome to Online IDE!! Happy Coding :)
[COLUMN 1: FREE, COLUMN 2: FREE, COLUMN 3: FREE, COLUMN 4: FREE, COLUMN 5: FREE, COLUMN 6: FREE, COLUMN 7: FREE]
Which column would you like to insert the coin
8
Which column would you like to insert the coin
6
```

Unexpected runtime error

```
Player 1 has dropped the Yellow chip
This is the column chip placed in:6
Exception in thread "main"
java.lang.NullPointerException
    at connectFour.checkAvailability(Main.java:335)
    at Player1.insertYellowChip(Main.java:457)
    at Player1.<init>(Main.java:438)
    at connectFour.<init>(Main.java:34)
    at Main.main(Main.java:408)
```

*** CODE ***

```
//********** OUTPUT      **********

//It is failing almost instantly in trying to access the board… I have commented certain areas of code also (which are highlighted in red), since it causes the execution to fail altogether of the software.
//I am unsure of the cause…


//Validation is ok for chip insertion


//Unexpected runtime error




//*** CODE ***


/*
```

Online Java - IDE, Code Editor, Compiler

Online Java is a quick and easy tool that helps you to build, compile, test your programs online.
*/

```
import java.util.Scanner;
import java.io.*;
import java.util.*;
import java.util.Arrays;

interface playConnectFour
{
   public void insertChip();
   public boolean checkAvailability(int input, int colour, String name);
   public boolean checkConnectFour(int colour);
   public void viewBoard();
}

class connectFour implements playConnectFour
{
   int [][] board;
   //int rows = 6;
   //int columns = 7;
   int j;
   int input;

   String availableColumns[] = {"COLUMN 1: FREE","COLUMN 2: FREE","COLUMN 3: FREE","COLUMN 4: FREE","COLUMN 5: FREE","COLUMN 6: FREE","COLUMN 7: FREE"};

   //THIS IS FAILING.. THIS IS TO LET END USER KNOW THE STATUS OF THE COLUMNS BEFORE PLACING A CHIP IN
   //System.out.println(Arrays.toString(availableColumns));


   Player1 p1 = new Player1(this, board);


   public connectFour(int[][] board)
   {
      this.board=board;

   }

   public void insertChip()
   {

   }


   public void viewBoard()
   {
      for (int [] temp: board)
      {
         System.out.println(Arrays.toString(temp));
```

```
    }

  }


  public boolean checkConnectFour(int colour)
  {
    //this will have to check if 4 in a row.

    // this will check first in the vertical direction

    int vertical=0;
    int horizontal=0;
    int diagonal=0;
    int counter=0;
    int upRow=0;
    int downRow=0;

    // CHECKING VERTICALLY

    // need logic here that if there is not a vertical yellow, it will set count back to 0
    // not so easy since it needs to count upwards and downwards

    if (j!=0)
    {
    for (int k = j-1; k>=0; k--)  // this is checking all rows below (same column) for yellows
    // since this does a pre-decrement, it would fail if j row = 0;

    {
      if (board[k][input]==colour)  //there is also a yellow in that position
      {
        vertical++;

      }

      if (board[k][input]==0)  // this is problem part of code... since it might find non-
matching colour below...
      // but vertically above there might be matching.....
      // so need to store counter in another variable potentially.
      {
        counter=vertical;
        vertical=0;


      }
    }
    }

    // row index is 0-5

    if (j!=5)
    {
     for (int k = j+1; k<=5; k++)  // this is checking all rows above (same column) for yellows
```

```
        {
           if (board[k][input]==colour)  //there is also a yellow in that position
           {
              vertical++;

           }

           if (board[k][input]==0)
           {
              counter=counter+vertical;
              vertical=0;

           }

        }
        }

        if (counter==3)  // this is 3 since it excludes the chip at [j][input]
        {
           System.out.println("connect 4");
           viewBoard();
           return true;
        }


        // CHECKING HORIZONTALLY
        // WITH HORIZONTAL CHECK, NEED TO CHECK EACH LAYER
        // SO NEED ANOTHER FOR LOOP


        // to the right

           counter=0;

        if (input!=6)  // Note the input has been set to 6 since can not check horizontally on 0-6
index notation (7 columns)
        {

        for (int j=0; j<=5;j++)  // this is to check each row index 0-5
        {
           counter=0;

        for (int m = input+1; m<=6; m++)  // this is checking the same row to the right for yellows..
        // it is adding 1 to colsRight to ensure 5 cols are checked
        {
           if (board[j][m]==colour)  //there is also a yellow in that position
           {
              horizontal++;

           }

           if (board[j][m]==0)
           {
              counter=horizontal;
```

```
            horizontal=0;

          }

       }
       }

   }


    if (input!=0)  // Note the input has been set to 0 since can not check to any columns to the left if chip entered in first column
       {

       // to the left

        for (int j=0; j<=5;j++)    // this is to check each row index 0-5
       {

        for (int m = input-1; m>=0; m--)  // this is checking all rows below (same column) for yellows
       {
          if (board[j][m]==colour)  //there is also a yellow in that position
          {
             horizontal++;

          }

          if (board[j][m]==0)
          {
             counter=counter+horizontal;
             horizontal=0;
          }

       }
       }
       }


       if (counter==3)
       {
          System.out.println("connect 4");
          viewBoard();
          return true;
       }



   counter = 0;


// CHECKING DIAGONALLY.

       // can not check diagonally upwards if the chip sits in top row or last column
```

```
      if (input!=6 &&  j!=5)
      {
      //this is checking diagonal with positive gradient
      for (int m = input+1; m<=6; m++)   // this has to reduce columns by 1 to ensure no overrun
diagonal check
      {
         upRow++;
         if (board[j+upRow][m]==colour)  //there is also a yellow in that position
         {
            diagonal++;

         }

         if (board[j+upRow][m]==0)
         {
            counter=diagonal;
            diagonal=0;

         }
      }
      }


      // this has to ensure that the chip is not sitting along the bottom line
      //this is checking diagonal with positive gradient. Extending the line.
      downRow = 0;


      if (j!=0  && input!=0)
      {

         for (int m = input-1; m>=0; m--)
         {
         downRow++;

         if (board[j-downRow][m]==colour)  //there is also a yellow in that position
         {
            diagonal++;

         }

         if (board[j-downRow][m]==0)
         {
            counter=counter + diagonal;
            diagonal=0;

         }
         }
   }

          if (counter==3)
      {
         System.out.println("connect 4");
         viewBoard();
```

```
        return true;
    }


    counter=0;
    upRow=0;


    //-------------------------------------------------
    //this is checking diagonal with negative gradient moving upwards

    if (input!=0 && j!=5)
    {
    for (int m = input-1; m<=0; m--)   // this has to reduce columns by 1 to ensure no overrun
diagonal check
    {
        upRow++;
        if (board[j+upRow][m]==colour)  //there is also a yellow in that position
        {
            diagonal++;

        }

        if (board[j+upRow][m]==0)
        {
            counter=diagonal;
            diagonal=0;

        }
    }
    }

    //this is checking diagonal with negative gradient. Extending the line.
    downRow = 0;

        if (input!=6 && j!=0)
        {

        for (int m = input+1; m>=6; m++)  //
        {
        downRow++;

        if (board[j-downRow][m]==colour)  //there is also a yellow in that position
        {
            diagonal++;

        }

        if (board[j-downRow][m]==0)
        {
            counter=counter+ diagonal;
            diagonal=0;
```

```
        }
        }
        }


        if (counter==3)
      {
        System.out.println("connect 4");
        viewBoard();

        return true;
      }


      return false;


}

   public boolean checkAvailability(int input, int colour, String name)
   {

      int instances=0;
      this.input=input;

      String chipColour="";
      boolean availability=false;

      //yellow is being assigned value 1

      switch(colour)
      {
        case 1:
          chipColour="Yellow";
          break;

        case 2:
          chipColour="Red";
          break;

          default:


      }

      System.out.println(name + " has dropped the " + chipColour + " chip");
      System.out.println("This is the column chip placed in:" + input);


      for (j=0; j<7;j++)
      {

        // the coin will drop to lowest position on the grid
```

```
        if (board[j][input]==0)   // all default values in declared array is 0
        {
           board[j][input]=colour;  // this will input a 1 to denote yellow if available.  2 to
denote red
           availability=true;

           viewBoard();
           checkConnectFour(colour);
           break;

        }

     }

     if (availability)
     {
        return true;

     }


     /*
     // This bit of code is not compiling. Logic seems correct.
     Intention is to mark the string array available columns with FULL if the column is full.
     // It will then count if instances is equal to number of columns... and if all columns full,
     //terminate the application.

     if (!availability)   // end user will be prompted to enter chip in different column if no
availability
     {
        availableColumns[input]="COLUMN " + input+": " + "FULL";

        for (String s:availableColumns)
        {
           if (s.indexof("FULL"))
           {
              instances++;
              if (instances==columns)
              {
                 System.out.println("The grid is all full");
                 System.exit(0);

              }
           }

        }

        checkAvailability(input,colour,name);


     }
     */
     return false;
```

```
    }

}


public class Main
{
    public static void main(String[] args) {
    System.out.println("Welcome to Online IDE!! Happy Coding :)");

    //int [][] board = new int [5][6];   // this ensures  7 columns and 6 rows

    int [][] board = {    {0,0,0,0,0,0,0},
                          {0,0,0,0,0,0,0},
                          {0,0,0,0,0,0,0},
                          {0,0,0,0,0,0,0},
                          {0,0,0,0,0,0,0},
                          {0,0,0,0,0,0,0},
                         };


    connectFour cf = new connectFour(board);




    }

}

class Player1
{
    int[][] currentBoard;
    connectFour cf;
    chips yellow;
    String name = "Player 1";

    Player2 p2;

    enum chips
    {
        YELLOW, RED;
    }


    public Player1(connectFour cf, int[][] currentBoard)
    {
        this.currentBoard=currentBoard;
        this.cf=cf;
        chips yellow = chips.YELLOW;
        this.yellow=yellow;

        insertYellowChip(yellow);
        p2=new Player2(cf, currentBoard);   // do not need this object.. but it is here just incase
need to get instance of p1
```

```
   }


   public void insertYellowChip(chips yellow)
   {
      int selection;
      System.out.println(Arrays.toString(cf.availableColumns));  // This will show end user free
columns

      do
      {
      System.out.println("Which column would you like to insert the coin");
      Scanner scan = new Scanner(System.in);
      selection = scan.nextInt();
      } while (selection<1 || selection>7);

      selection = selection - 1;       //1 has been subtracted since it is a zero index notation
board....

      cf.checkAvailability(selection,1,name);
   }


   public void insertYellowChip()
   {
      insertYellowChip(yellow);
   }


}


class Player2
{
   int[][]currentBoard;
   connectFour cf;
   String name = "Player 2";
   Player1 p1;

   enum chips
   {
      YELLOW, RED;
   }


   public Player2(connectFour cf, int[][] currentBoard)
   {
      this.cf=cf;
      this.currentBoard=currentBoard;
      chips red = chips.RED;
      insertRedChip(red);
      p1=new Player1(cf, currentBoard);   // do not need this object.. but it is here just incase
```

```
need to get instance of p1
   }

   public void insertRedChip (chips red)
   {
      int selection;
      System.out.println(Arrays.toString(cf.availableColumns));  // This will show end user free
columns

      do
      {
      System.out.println("Which column would you like to insert the coin");
      Scanner scan = new Scanner(System.in);
      selection = scan.nextInt();
      } while (selection<1 || selection>7);

      selection = selection - 1;       //1 has been subtracted since it is a zero index notation
board....

      cf.checkAvailability(selection,2,name);

   }


}
```