```
//FPModel.cs
using System;

public delegate void UpdatedEventHandler(object sender, EventArgs e);

namespace modelTest
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            FPModel model = new FPModel ();

            if (model.winningLine (1) == true) {
                Console.WriteLine ("Winning line found");
            } else {
                Console.WriteLine ("Winning line NOT found");
            }

            Console.WriteLine ("setting pieces...");
            model.setPiece (0, 0, 1);
            model.setPiece (0, 1, 1);
            model.setPiece (0, 2, 1);
            model.setPiece (0, 3, 1);
            Console.WriteLine ("pieces set!");

            if (model.winningLine (1) == true) {
                Console.WriteLine ("Winning line found");
            } else {
                Console.WriteLine ("Winning line NOT found");
            }

            Console.ReadLine();
        }
    }

    public class FPModel
    {

        public event UpdatedEventHandler Updated;

        protected virtual void OnUpdated(EventArgs e)
        {

            if (Updated != null)
            {
                this.Updated(this, e);
            }
        }

        private static int noOfColomns = 7;
        private static int noOfRows = 6;
        private int[][] boardStatus = new int[noOfRows][];
        private int player1, player2;
        private bool boardEmpty = true;
```

```
public FPModel()
{
    player1 = 0;
    player2 = 0;
    for (int x = 0; x < boardStatus.Length; x++)
    {
        boardStatus[x] = new int[noOfColomns];
    }
    clearBoard();
}

public void clearBoard()
{
    for (int i = 0; i < noOfRows; i++)
    {
        for (int j = 0; j < noOfColomns; j++)
        {
            boardStatus[i][j] = 0;
        }
    }
    boardEmpty = true;
    this.OnUpdated(new  EventArgs());
}

public int NoOfColomns
{
    get{return noOfColomns;}
}


public int NoOfRows
{
    get{ return noOfRows;}
}


public int[][] ChipStatus
{
    get{ return boardStatus; }
}


public int getScore(int player)
{
    switch (player)
    {
    case 1:
        return player1;
    case 2:
        return player2;
    default:
        return 0;
    }
}


public void setScore(int player, int score)
```

```
        {
            if (player == 1)
            {
                player1 = score;
            }
            if (player == 2)
            {
                player2 = score;
            }
            this.OnUpdated(new  EventArgs());
        }

        public bool validMove(int row, int col)
        {
            if(boardStatus[row][col]==0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }


        public void setPiece(int colomn, int value)
        {
            if (boardEmpty == true)
            {
                boardEmpty = false;
            }

            for(int i =0 ; i < noOfRows; i++){
                if(boardStatus[i][colomn]==0){
                    boardStatus[i][colomn]=value;
                    break;
                }
            }
            this.OnUpdated(new  EventArgs());
        }


        public bool boardIsEmpty()
        {
            return boardEmpty;
        }

        public bool winningLine(int player)
        {
            for(int row = 0; row < noOfRows;row++)
            {
                for(int col = 0; col < noOfColomns; col++)
                {
                    if(hasNeighbour(1,1,row,col,player) >=4)
                        return true;
                    if(hasNeighbour(1,0,row,col,player) >=4)
                        return true;
```

```
                    if(hasNeighbour(0,1,row,col,player) >=4)
                        return true;
                    if(hasNeighbour(1,-1,row,col,player) >=4)
                        return true;
                }
            }
            return false;
        }

        private int hasNeighbour(int xDir,int yDir,int row, int col, int player)
        {
            int found=0;
            if((row>=noOfRows||row<0)||(col>=noOfColomns||col<0))
                return 0;
            if(boardStatus[row][col]==player)
            {
                found=1;
                if((xDir == 1)&&(yDir == 1))
                {
                    //Up diagonal search
                    return found+hasNeighbour(xDir,yDir,row+1,col+1,player);
                }
                if((xDir == 1)&&(yDir == 0))
                {
                    //Horizontal search
                    return found+hasNeighbour(xDir,yDir,row,col+1,player);
                }
                if((xDir == 0)&&(yDir == 1))
                {
                    //Vertical search
                    return found+hasNeighbour(xDir,yDir,row+1,col,player);
                }
                if((xDir == 1)&&(yDir == -1))
                {
                    //Down diagonal search
                    return found+hasNeighbour(xDir,yDir,row-1,col+1,player);
                }
                return 0;
            }else
            {
                return 0;
            }
        }

    }
}
```