~\OneDrive - St Paul's Catholic College\Documents\2D Strategy Game - Liberator\Assets\Scripts\NeighboursFinder.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;
using Unity.VisualScripting;

public class NeighboursFinder : MonoBehaviour
{
    private HexData startingHex; // the hex on where our hero is standing on
    static List<HexData> allNeighbours = new List<HexData>(); // list containing all the hexes neighbouring the starting position aka where the soldier is standing on

    static public List<HexData> GetAdjacentHexes(HexData startingHex, EvaluateHex checkHex) // looks for neighbouring hexes
    {
        allNeighbours.Clear(); // clearing the list before assigning a new one
        // to get the array index from the hex coordinate need to subtract 1. For example the first coordinate hex (1,1) is [0,0] in the allHexesArray so we -1 on x and y
        int initialX = startingHex.horizontalCoordinate -1; // first index for two dimensional list
        int initialY = startingHex.verticalCoordinate -1; // second index for two dimensional list

        // iterates x and y from -1 to 1 to get adjacent hexes referring to the coordinates of starting hex
        for (int x = - 1; x <= 1; x++) // when x is = -1, y = -1,0,1 then it goes to x = 0, y = -1,0,1 again etc
        {
         for (int y = -1; y <= 1; y++)
         {
            // checking if the hexes are within the bounds 16 rows and 19 hexes in each row
            if (initialX + x >= 0 &&  initialY + y >= 0 && initialX + x < 19 && initialY + y < 16)
            {
                // checking if hexes are valid to move on and making sure the starting hex isn't included for the player to move on
                if (x + y != 0 && checkHex.EvaluateHex(FieldManager.allHexesArray[initialX + x, initialY +y]))
                {
                    allNeighbours.Add(FieldManager.allHexesArray[initialX + x, initialY + y]); // adding the hexes to the allneighbours list
                }
            }
         }
       }
       return allNeighbours; // returning the list
    }

   /* private bool inBounds(int index, HexData[] array)
    {
        return (index >= 0) && (index < array.Length);
    }*/
}

```