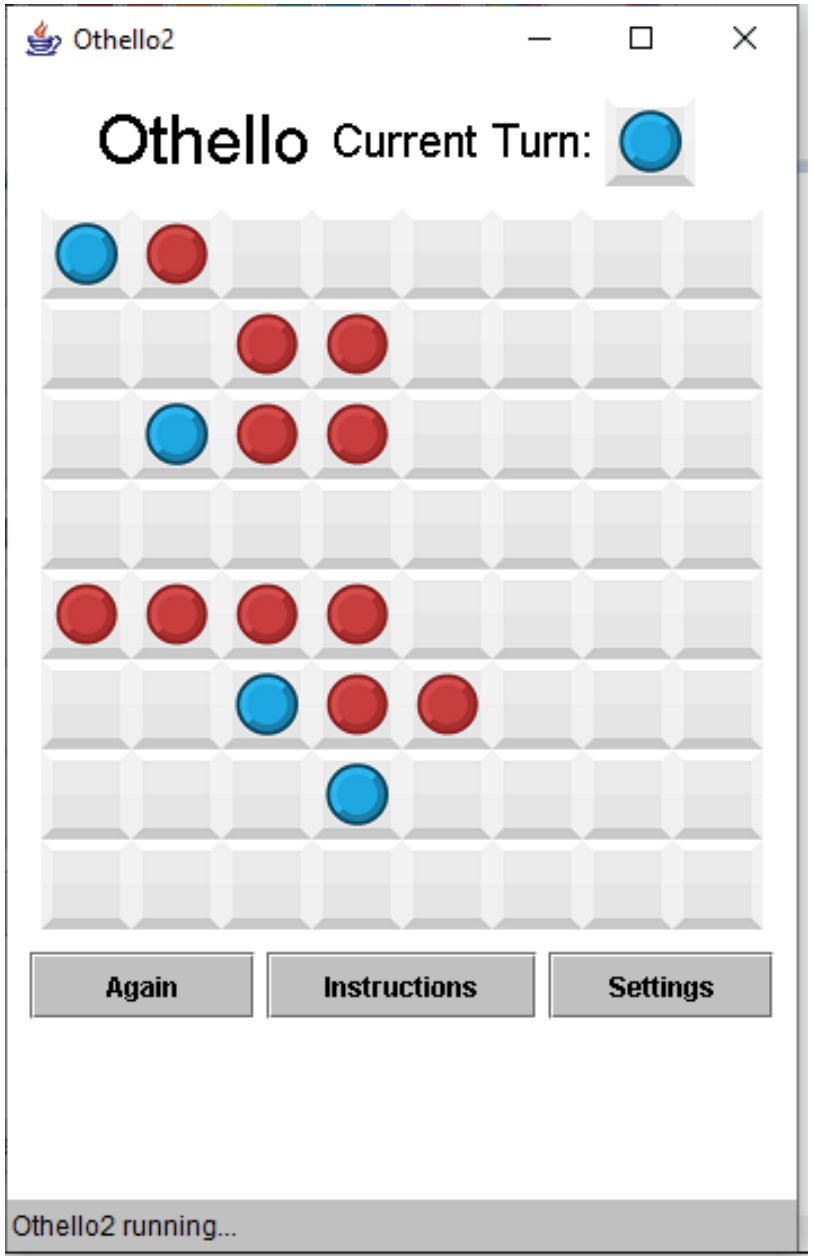
Othello Movement

canGo Right

Set Up Board to Test Well

```
//Game screen
JLabel turnPic;
int turn = 1;
//grid
int row = 8;
int col = 8;
JButton a[] = new JButton [row * col];
int b[] [] = {{1, 2, 0, 0, 0, 0, 0, 0},
              {0, 0, 2, 2, 0, 0, 0, 0},
              {0, 1, 2, 2, 0, 0, 0, 0},
              {0, 0, 0, 0, 0, 0, 0, 0},
              {2, 2, 2, 2, 0, 0, 0, 0},
              {0, 0, 1, 2, 2, 0, 0, 0},
              {0, 0, 0, 1, 0, 0, 0, 0},
              {0, 0, 0, 0, 0, 0, 0, 0}};
```



In ActionPerformed

```
//TO DO: Fill this comment in
else
{
    int n = Integer.parseInt (e.getActionCommand ());
    int x = n / col;
    int y = n % col;
    showStatus ("" + canGo (x, y));
    if (canGo (x, y))
        move (x, y);
    else
        showStatus ("Sorry, You can't move there.");
    redraw ();

}
```

CanGo Method

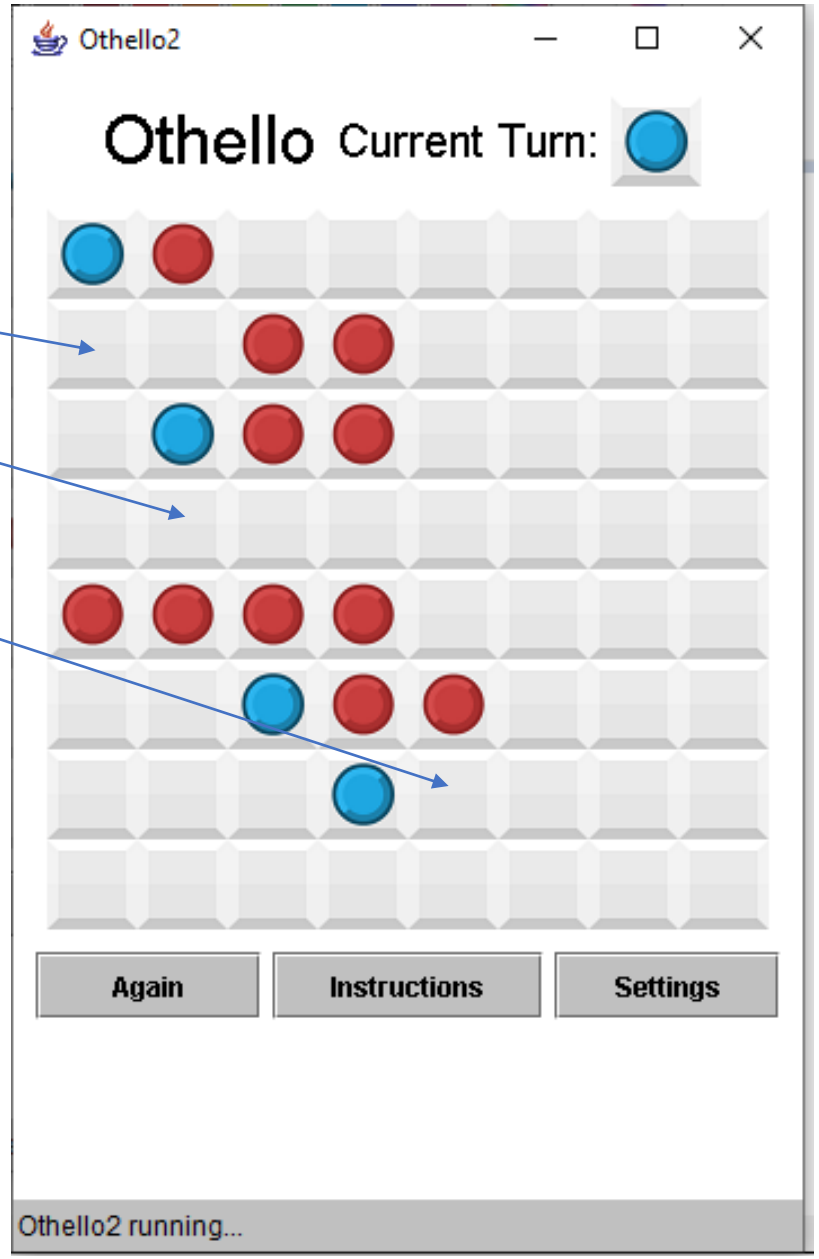
```
public boolean canGo (int x, int y)
{ //This checks if a turn is valid
     if (b [x] [y] != 0)
          return false;
     else if (canGoLeft (x, y) == true)
          return true;

     //TO DO: other directions here

     else
          return false;
}
```

Move Method

```
public void move (int x, int y)
{ //Place the piece, swap the middle ones.
      b [x] [y] = turn;

      if (canGoLeft (x, y))
      {
            swapLeft (x, y);
      }
}
```

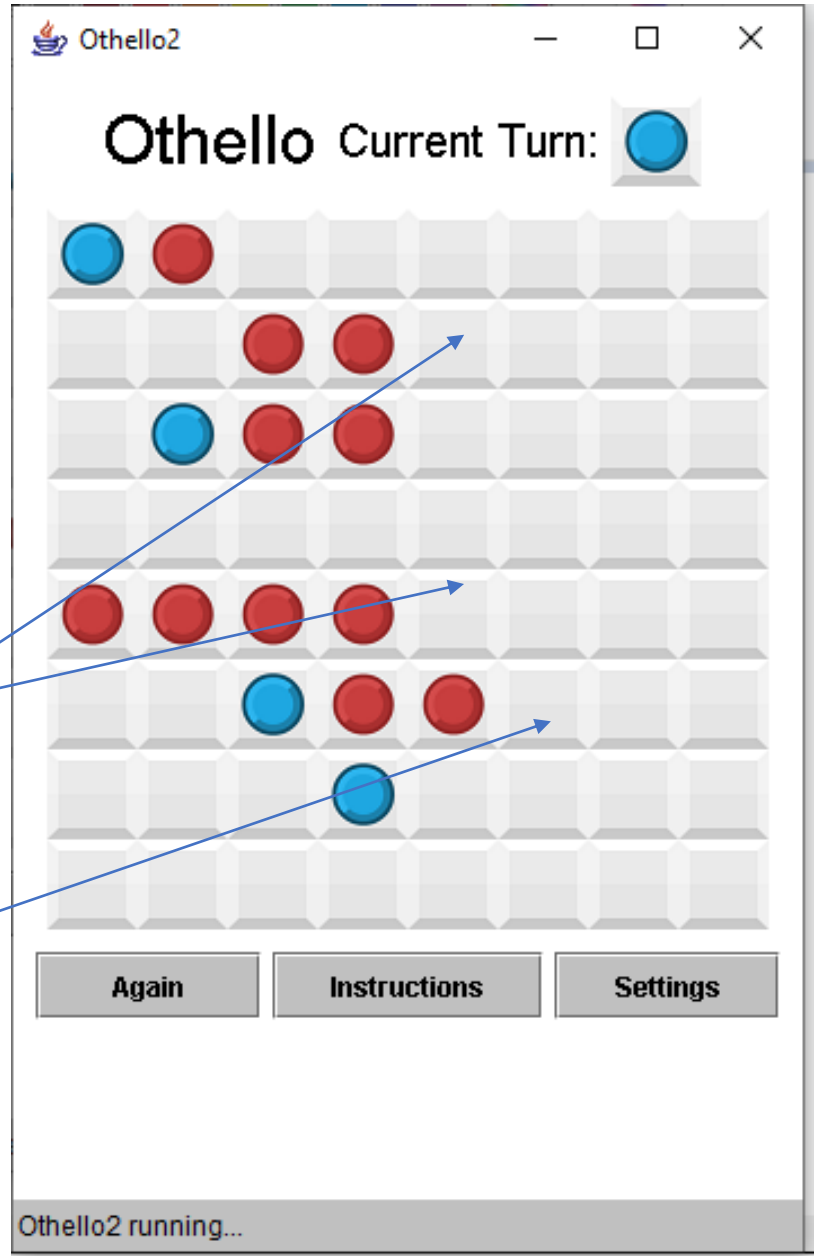
Can Go Left

```
public boolean canGoLeft (int x, int y)
{ // Checks if a player can go in (x,y) based on it's left side
    int me = turn;
    int them = 1;
    if (turn == 1)
        them = 2;

    //at edge, can't go
    if (y - 1 < 0)
        return false;
    //nothing to left, can't go
    else if (y - 1 >= 0 && b [x] [y - 1] == 0)
        return false;
    //my piece to left, can't go
    else if (y - 1 >= 0 && b [x] [y - 1] == me)
        return false;
    //them to left, need to check further left
    else
    {
        int ycopy = y - 1;
        while (ycopy >= 0 && b [x] [ycopy] == them)
        {
            ycopy--;
        }
        //them all the way to the edge
        if (ycopy < 0)
            return false;
        //them all the way to a blank
        else if (ycopy >= 0 && b [x] [ycopy] == 0)
            return false;
        //them all the way to me
        else if (ycopy >= 0 && b [x] [ycopy] == me)
            return true;

    }
    return false;
}
```



Can Go Left 2

```
public boolean canGoLeft (int x, int y)
{ // Checks if a player can go in (x,y) based on it's left side
    int me = turn;
    int them = 1;
    if (turn == 1)
        them = 2;

    //at edge, can't go
    if (y - 1 < 0)
        return false;
    //nothing to left, can't go
    else if (y - 1 >= 0 && b [x] [y - 1] == 0)
        return false;
    //my piece to left, can't go
    else if (y - 1 >= 0 && b [x] [y - 1] == me)
        return false;
    //them to left, need to check further left
    else
    {
        int ycopy = y - 1;
        while (ycopy >= 0 && b [x] [ycopy] == them)
        {
            ycopy--;
        }
        //them all the way to the edge
        if (ycopy < 0)
            return false;
        //them all the way to a blank
        else if (ycopy >= 0 && b [x] [ycopy] == 0)
            return false;
        //them all the way to me
        else if (ycopy >= 0 && b [x] [ycopy] == me)
            return true;

    }
    return false;
}
```



Swap Left

```
public void swapLeft (int x, int y)
{// This swaps the pieces on the left side
     int me = turn;
     int them = 1;
     if (turn == 1)
          them = 2;

     int ycopy = y - 1;
     while (ycopy >= 0 && b [x] [ycopy] == them)
     {
          b [x] [ycopy] = me;
          ycopy--;
     }
}
```