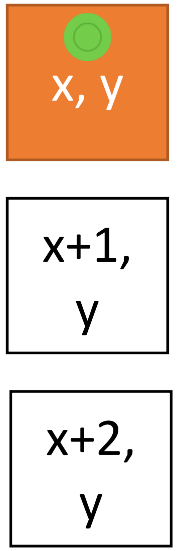
Othello

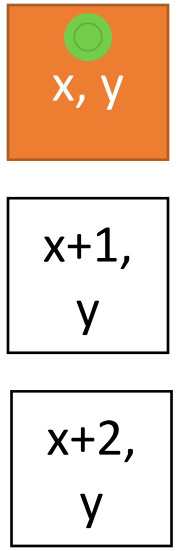
Two More Directions: Up and UpRight

Down



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

    int xcopy = x + 1;
    while (xcopy < row && b [xcopy] [y] == them)
    {
        b [xcopy] [y] = me;
        xcopy++;
    }
}
```

Down



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

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

    }
    return false;
}
```

Down

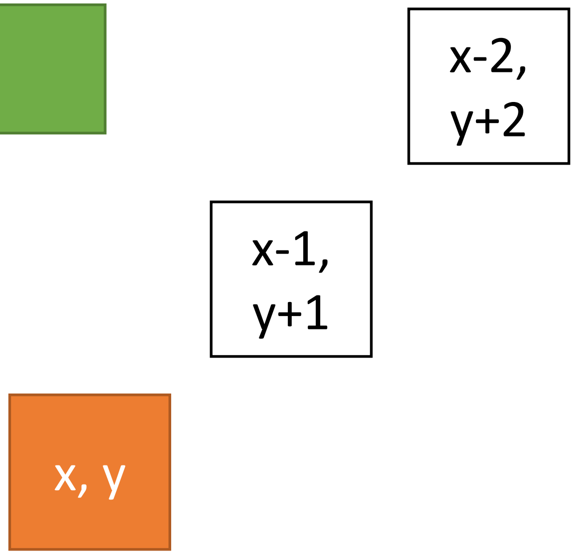
```
public void move (int x, int y)
{ //Place the piece, swap the middle ones.
    b [x] [y] = turn;

    if (canGoLeft (x, y))
        swapLeft (x, y);
    if (canGoRight (x, y))
        swapRight (x, y);
    if (canGoDown (x, y))
        swapDown (x, y);
    if (canGoUp (x, y))
        swapUp (x, y);
    if (canGoUpRight (x, y))
        swapUpRight (x, y);
}
```

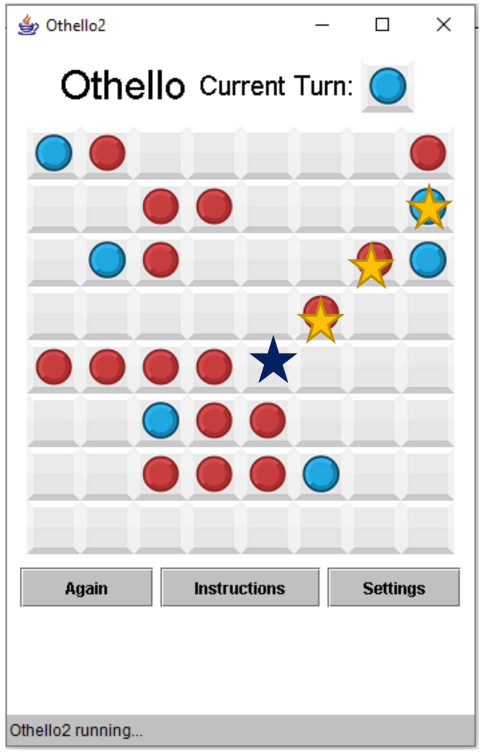
Down

```
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;
    else if (canGoRight (x, y) == true)
        return true;
    else if (canGoUp (x, y) == true)
        return true;
    else if (canGoDown (x, y) == true)
        return true;
    else if (canGoUpRight (x, y) == true)
        return true;
    //TO DO: other directions here

    else
        return false;
}
```

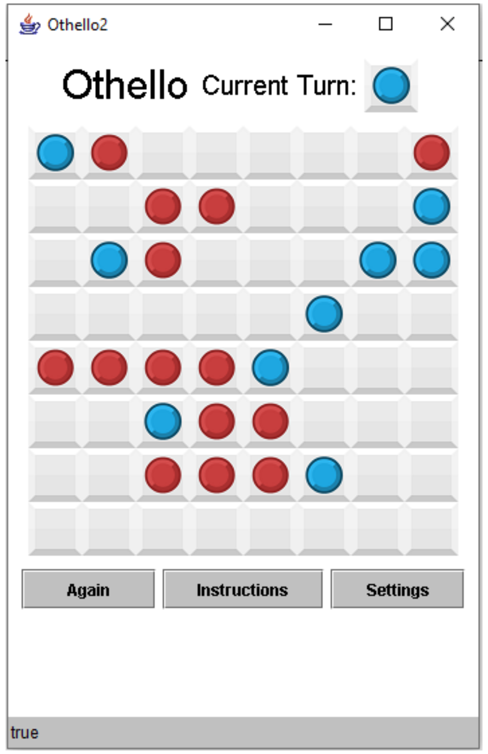
Up Right

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

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





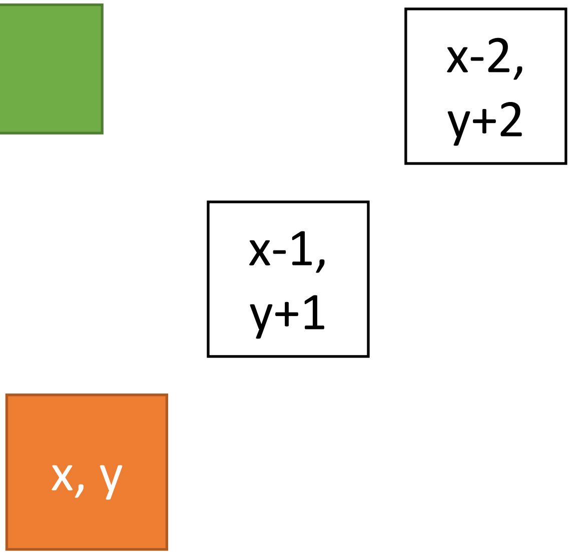
Othello2 running...

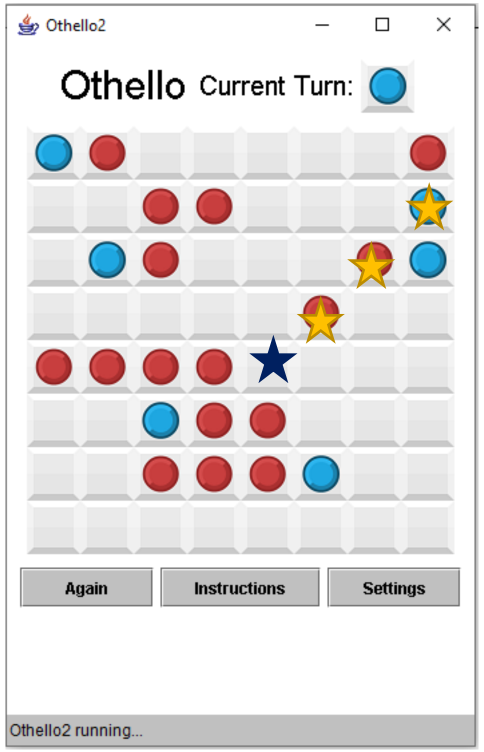


true

Up Right

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

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

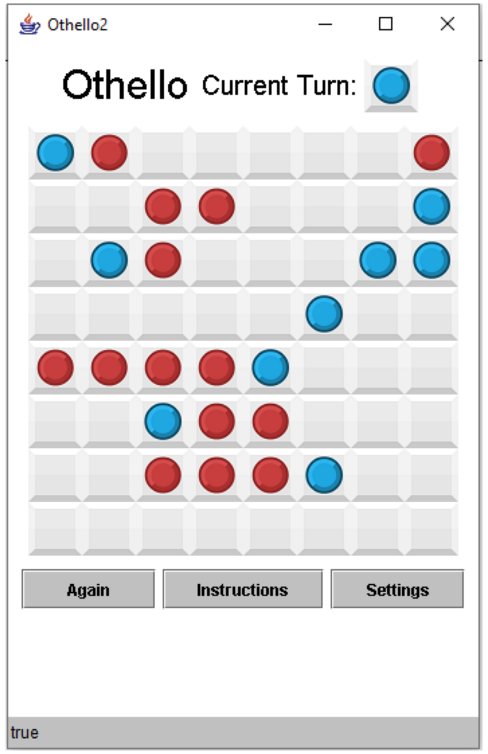



Othello2

Othello Current Turn:

Again Instructions Settings

Othello2 running...


Othello2

Othello Current Turn:

Again Instructions Settings

true

Swap Up Right Colour coded

```java
public void swapUpRight (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;
    int xcopy = x - 1;
    while (ycopy < col && xcopy >= 0 && b [xcopy] [ycopy] == them)
    {
        b [xcopy] [ycopy] = me;
        ycopy++;
        xcopy--;
    }
}
```

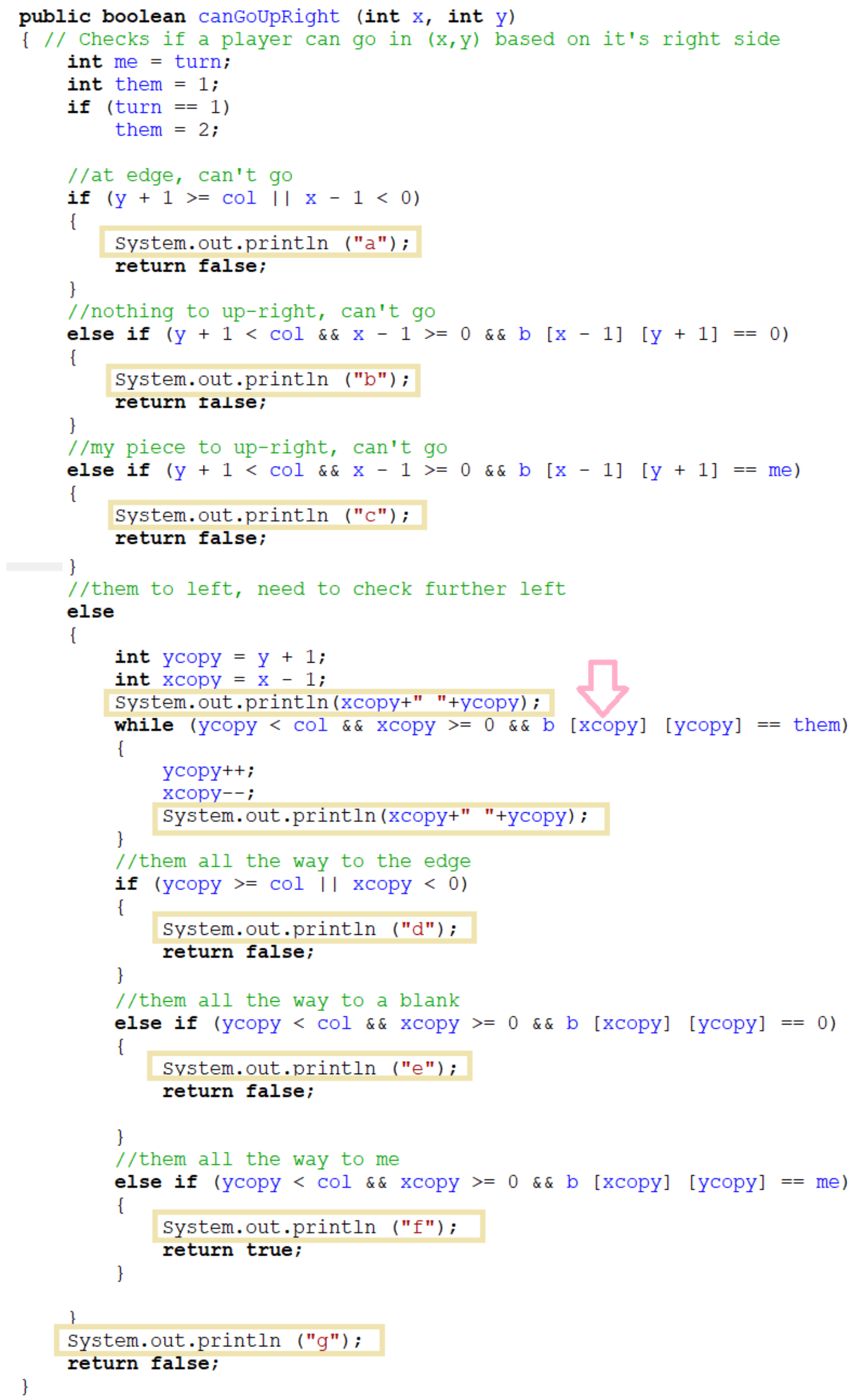
Swap Up Right

```java
public void swapUpRight (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;
    int xcopy = x - 1;
    while (ycopy < col && xcopy >= 0 && b [xcopy] [ycopy] == them)
    {
        b [xcopy] [ycopy] = me;
        ycopy++;
        xcopy--;
    }
}
```

```
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);
    }
    if (canGoRight (x, y))
    {
        swapRight (x, y);
    }
    if (canGoUp (x, y))
    {
        swapUp (x, y);
    }

    if (canGoUpRight (x, y))
    {
        swapUpRight (x, y);
    }
}
```

```
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;
    else if (canGoRight (x, y) == true)
        return true;
    else if (canGoUp (x, y) == true)
        return true;
    else if (canGoUpRight (x, y) == true)
        return true;
    //TO DO: other directions here

    else
        return false;
}
```

Error Testing



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

    //at edge, can't go
    if (y + 1 >= col || x - 1 < 0)
    {
        System.out.println ("a");
        return false;
    }
    //nothing to up-right, can't go
    else if (y + 1 < col && x - 1 >= 0 && b [x - 1] [y + 1] == 0)
    {
        System.out.println ("b");
        return false;
    }
    //my piece to up-right, can't go
    else if (y + 1 < col && x - 1 >= 0 && b [x - 1] [y + 1] == me)
    {
        System.out.println ("c");
        return false;
    }
    //them to left, need to check further left
    else
    {
        int ycopy = y + 1;
        int xcopy = x - 1;
        System.out.println(xcopy+" "+ycopy);
        while (ycopy < col && xcopy >= 0 && b [xcopy] [ycopy] == them)
        {
            ycopy++;
            xcopy--;
            System.out.println(xcopy+" "+ycopy);
        }
        //them all the way to the edge
        if (ycopy >= col || xcopy < 0)
        {
            System.out.println ("d");
            return false;
        }
        //them all the way to a blank
        else if (ycopy < col && xcopy >= 0 && b [xcopy] [ycopy] == 0)
        {
            System.out.println ("e");
            return false;

        }
        //them all the way to me
        else if (ycopy < col && xcopy >= 0 && b [xcopy] [ycopy] == me)
        {
            System.out.println ("f");
            return true;
        }

    }
    System.out.println ("g");
    return false;
}
```

Error Testing Close up 1

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

    //at edge, can't go
    if (y + 1 >= col || x - 1 < 0)
    {
        System.out.println ("a");
        return false;
    }
    //nothing to up-right, can't go
    else if (y + 1 < col && x - 1 >= 0 && b [x - 1] [y + 1] == 0)
    {
        System.out.println ("b");
        return false;
    }
    //my piece to up-right, can't go
    else if (y + 1 < col && x - 1 >= 0 && b [x - 1] [y + 1] == me)
    {
        System.out.println ("c");
        return false;
    }
```

Error Testing Close up 2

```
    //them to left, need to check further left
    else
    {
        int ycopy = y + 1;
        int xcopy = x - 1;
        System.out.println(xcopy+" "+ycopy);
        while (ycopy < col && xcopy >= 0 && b [xcopy] [ycopy] == them)
        {
            ycopy++;
            xcopy--;
            System.out.println(xcopy+" "+ycopy);
        }
        //them all the way to the edge
        if (ycopy >= col || xcopy < 0)
        {
            System.out.println ("d");
            return false;
        }
        //them all the way to a blank
        else if (ycopy < col && xcopy >= 0 && b [xcopy] [ycopy] == 0)
        {
            System.out.println ("e");
            return false;

        }
        //them all the way to me
        else if (ycopy < col && xcopy >= 0 && b [xcopy] [ycopy] == me)
        {
            System.out.println ("f");
            return true;
        }

    }
    System.out.println ("g");
    return false;
}
```