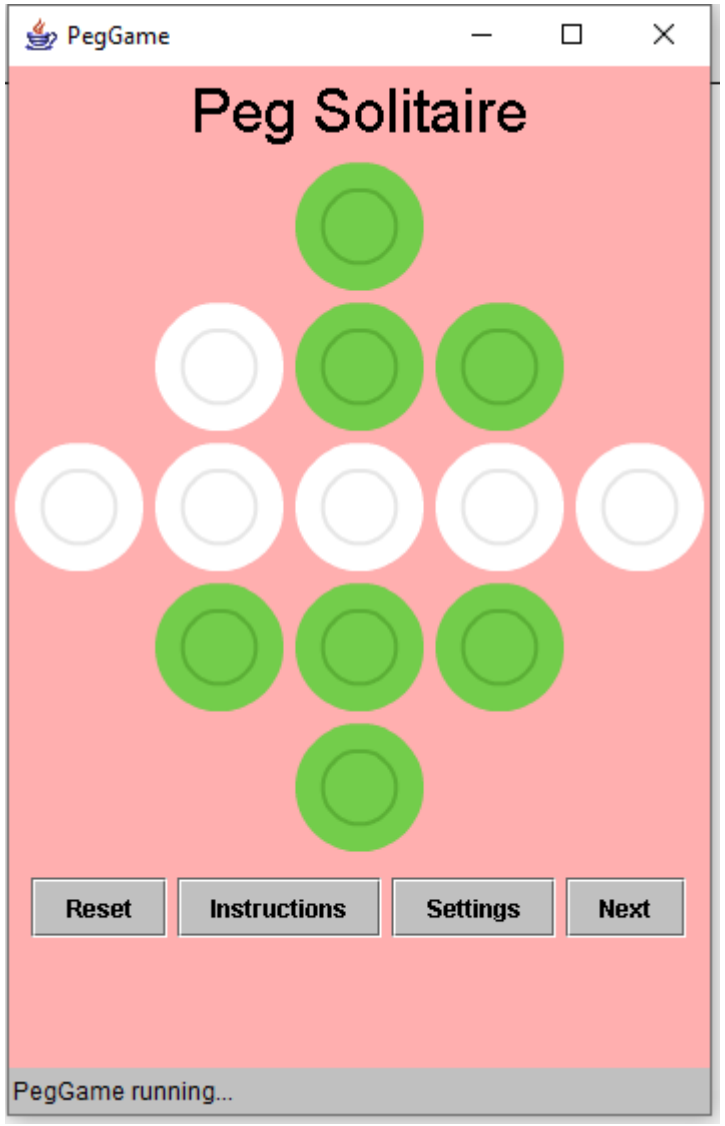
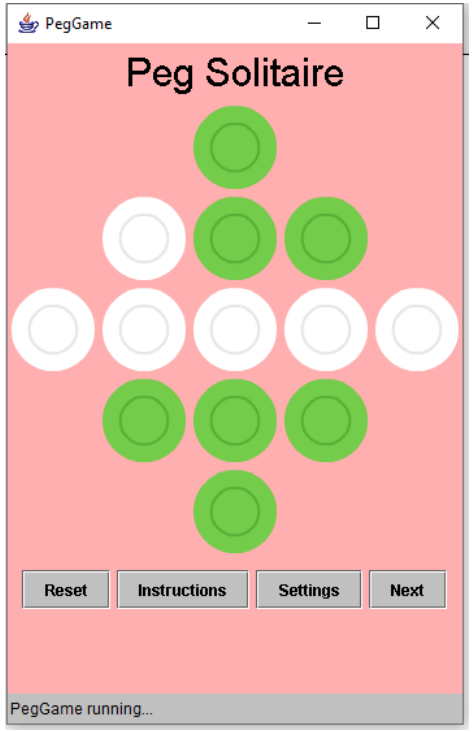


Peg Game

Jumping Movement

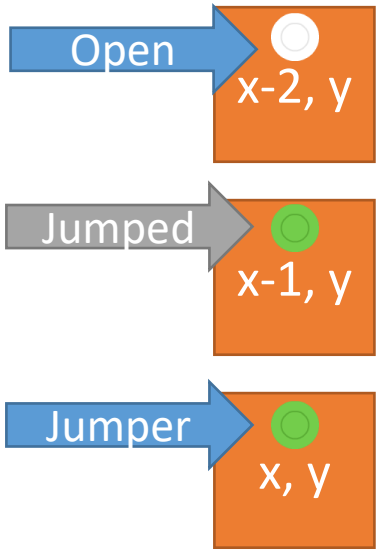
Global Variables

```
//Game screen
JLabel turnPic;
int turn = 1;
int last = -1;
//grid
int row = 5;
int col = 5;
JButton a[] = new JButton [row * col];

int b[] [] = {{2, 2, 1, 2, 2}, {2, 0, 1, 1, 2}, {0, 0, 0, 0, 0}, {2, 1, 1, 1, 2}, {2, 2, 1, 2, 2}};
int levelCount = 10;

//Formatting
Color backgroundColour = Color.pink;
Color buttonColour = Color.lightGray;
Color buttonText = Color.black;
Color titleColour = Color.black;
Font titleFont = new Font ("Arial", Font.PLAIN, 30);
Font promptFont = new Font ("Arial", Font.PLAIN, 20);
Dimension boardSquare = new Dimension (70, 70);
```



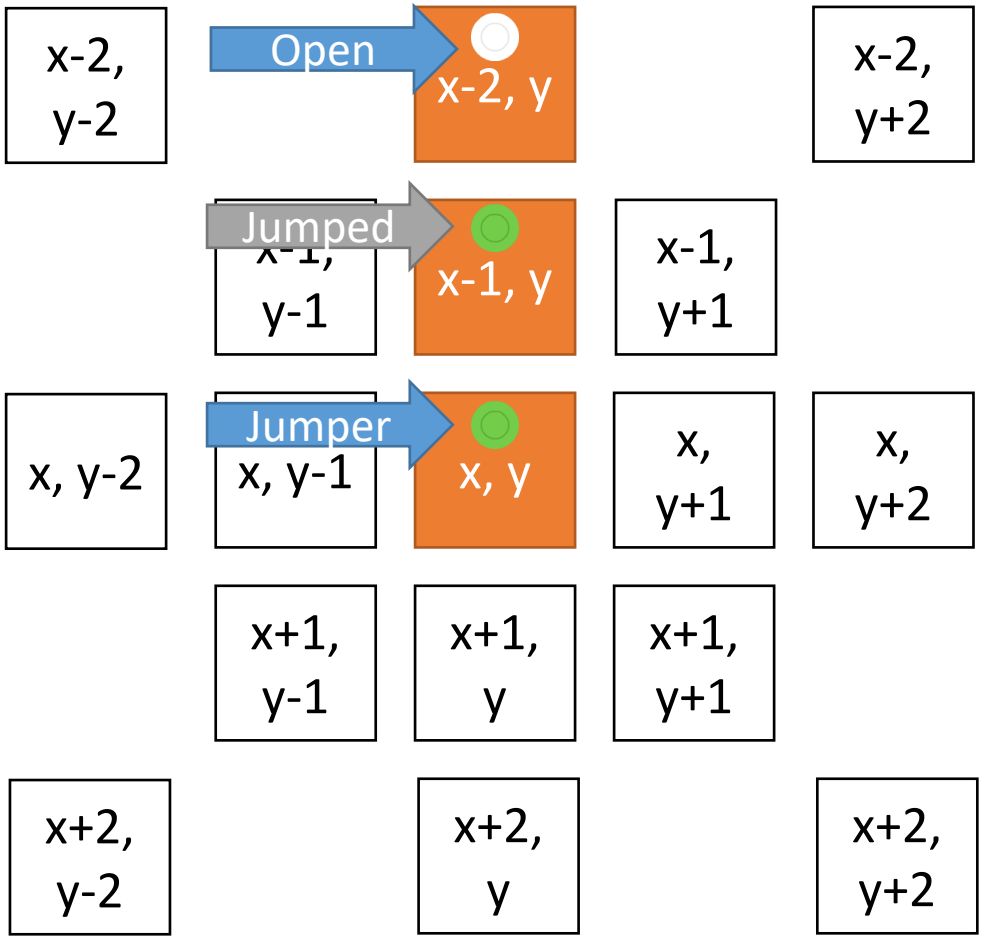
Two Click Process in ActionPerformed

```
//TO DO: Fill this comment in
else
{
    int n = Integer.parseInt (e.getActionCommand ());
    if (last == -1)
    {
        last = n;
    }
    else
    {
        if (isValidUp (n, last))
        {
            moveUp (n, last);
        }

        last = -1;
        //other checks go here
        redraw ();
    }

}
```

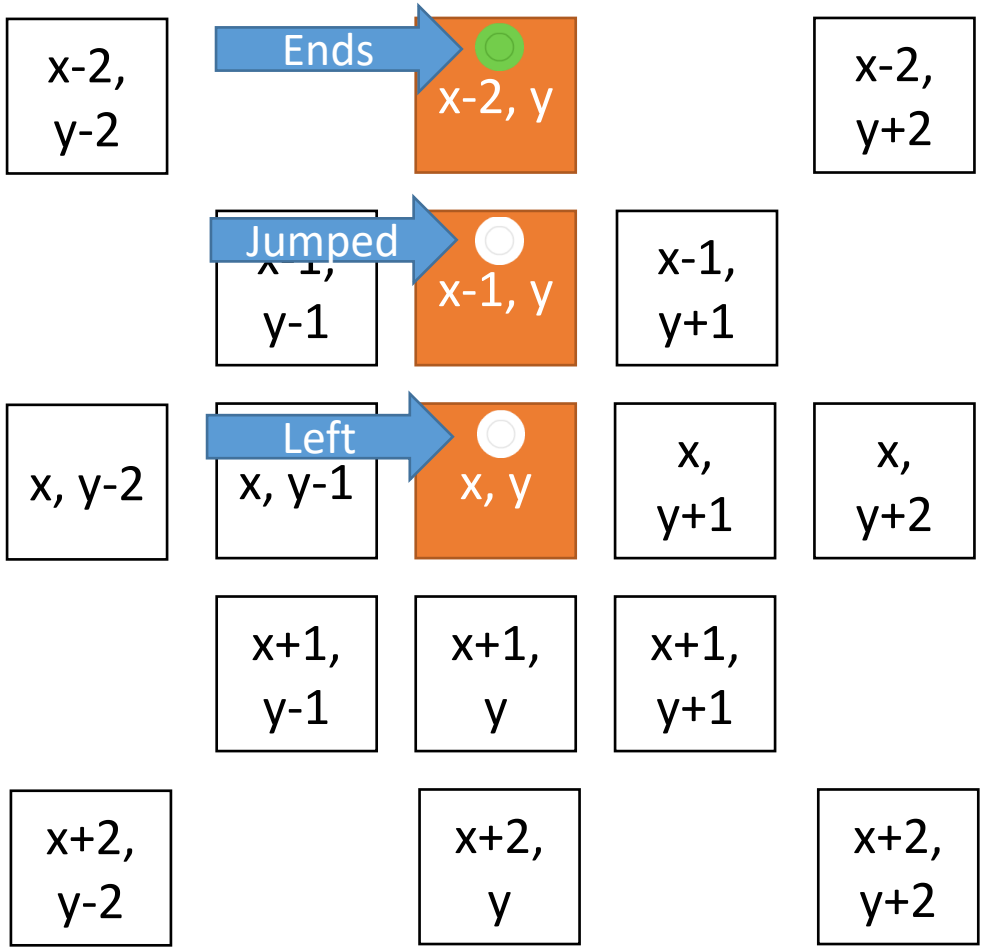


Up

Before

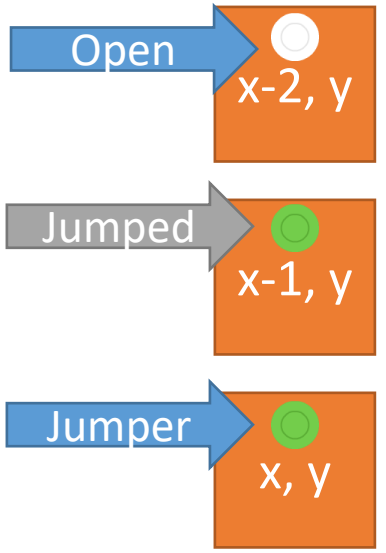


After



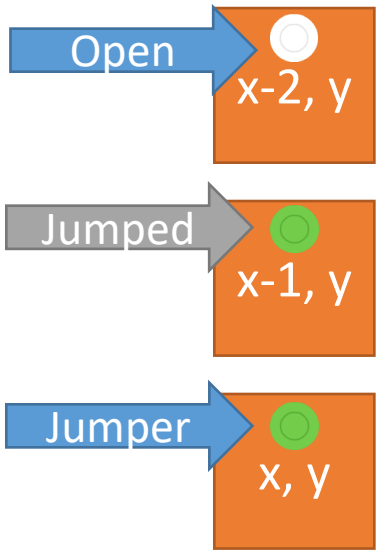
Can Go Up

```
public boolean isValidUp (int pos, int last)
{
    int endX = pos / col;
    int endY = pos % col;
    int startX = last / col;
    int startY = last % col;
    //end must be blank
    if (b [endX] [endY] != 0)
        return false;
    //start must be peg
    else if (b [startX] [startY] != 1)
        return false;
    //middle must be peg
    else if (startX - 1 >= 0 && b [startX - 1] [startY] != 1)
        return false;
    //start and end form correct relationship
    else if (startX - 2 >= 0 && (startX - 2) == endX && startY == endY)
        return true;
    //otherwise, it's all bad
    else
        return false;
}
```

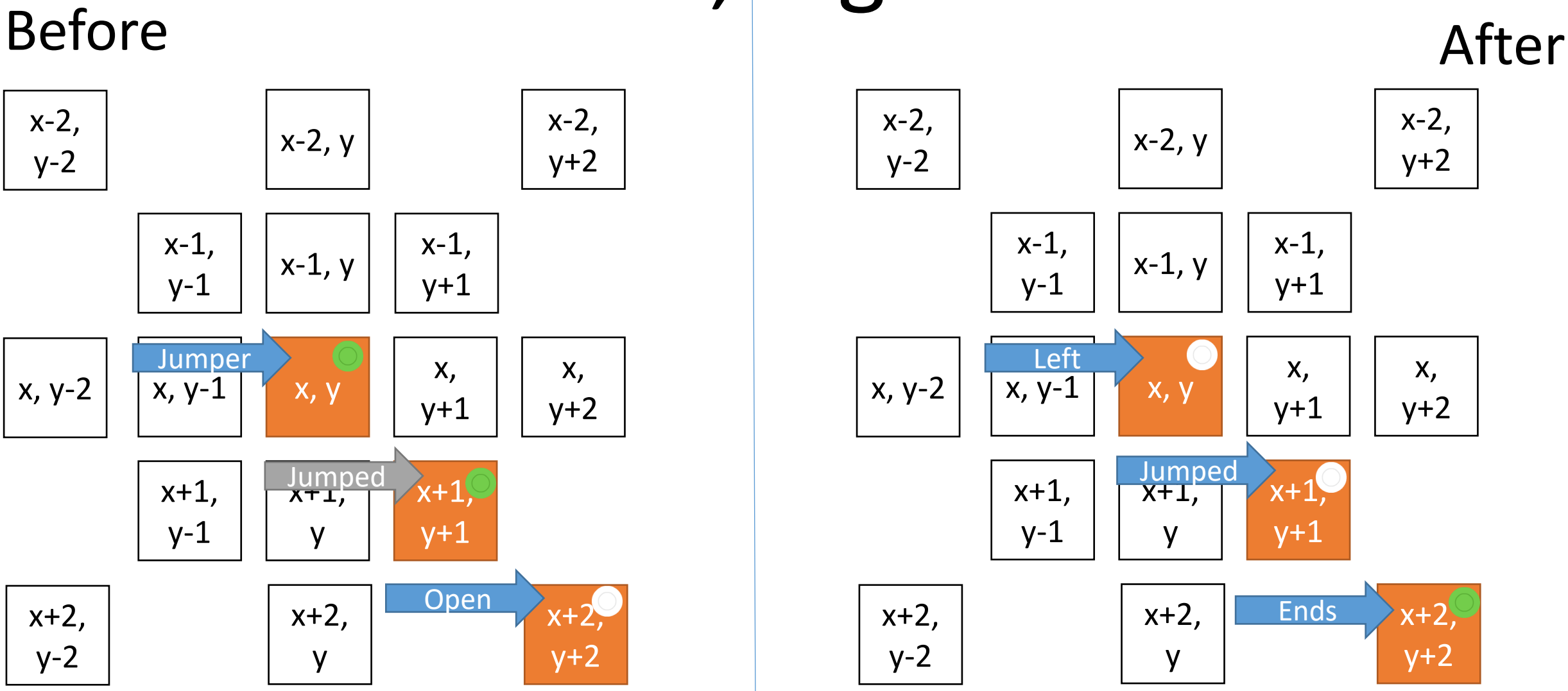


Can Go Up

```
public void moveUp (int pos, int last)
{
      int startX = last / col;
      int startY = last % col;
      int endX = pos / col;
      int endY = pos % col;
      b [startX] [startY] = 0;
      b [startX - 1] [startY] = 0;
      b [endX] [endY] = 1;
}
```

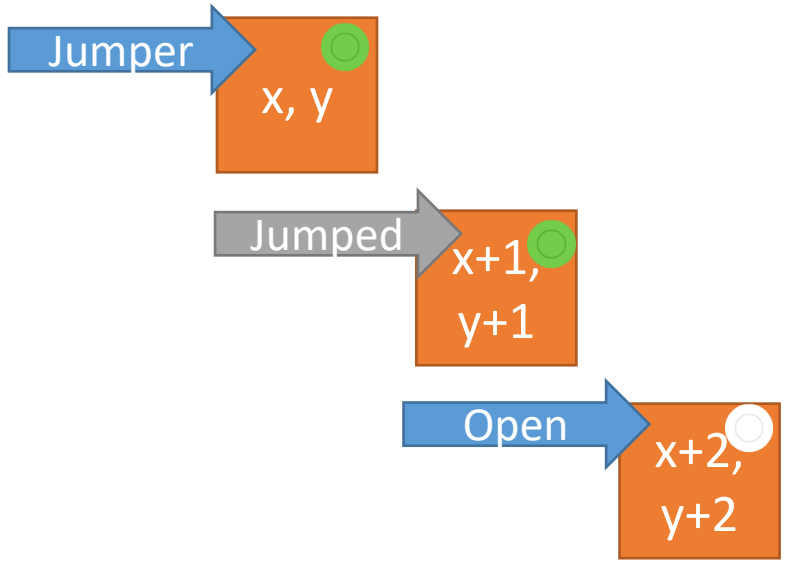


Down, Right


Before
x-2, y-2
x-2, y
x-2, y+2
x-1, y-1
x-1, y
x-1, y+1
x, y-2
Jumper
x, y-1
x, y
x, y+1
x, y+2
x+1, y-1
Jumped
x+1, y
x+1, y+1
x+2, y-2
x+2, y
Open
x+2, y+2
After
x-2, y-2
x-2, y
x-2, y+2
x-1, y-1
x-1, y
x-1, y+1
x, y-2
Left
x, y-1
x, y
x, y+1
x, y+2
x+1, y-1
Jumped
x+1, y
x+1, y+1
x+2, y-2
x+2, y
Ends
x+2, y+2

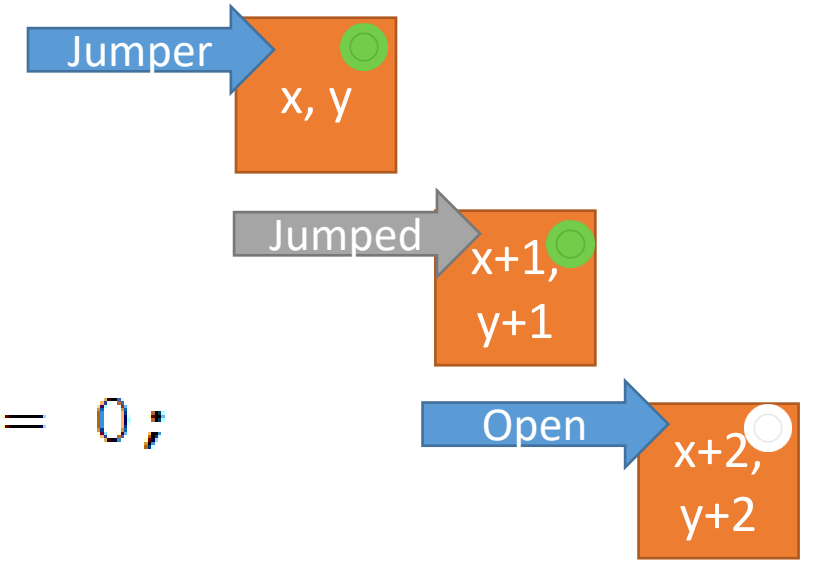
Can Go Down Right

```
public boolean isValidDownRight (int pos, int last)
{
    int endX = pos / col;
    int endY = pos % col;
    int startX = last / col;
    int startY = last % col;
    //end must be blank
    if (b [endX] [endY] != 0)
        return false;
    //start must be peg
    else if (b [startX] [startY] != 1)
        return false;
    //middle must be peg
    else if (startX + 1 < row && startY + 1 < col && b [startX + 1] [startY + 1] != 1)
        return false;
    //start and end form correct relationship
    else if (startX + 2 < row && startY + 2 < col && (startX + 2) == endX && (startY + 2) == endY)
        return true;
    //otherwise, it's all bad
    else
        return false;
}
```



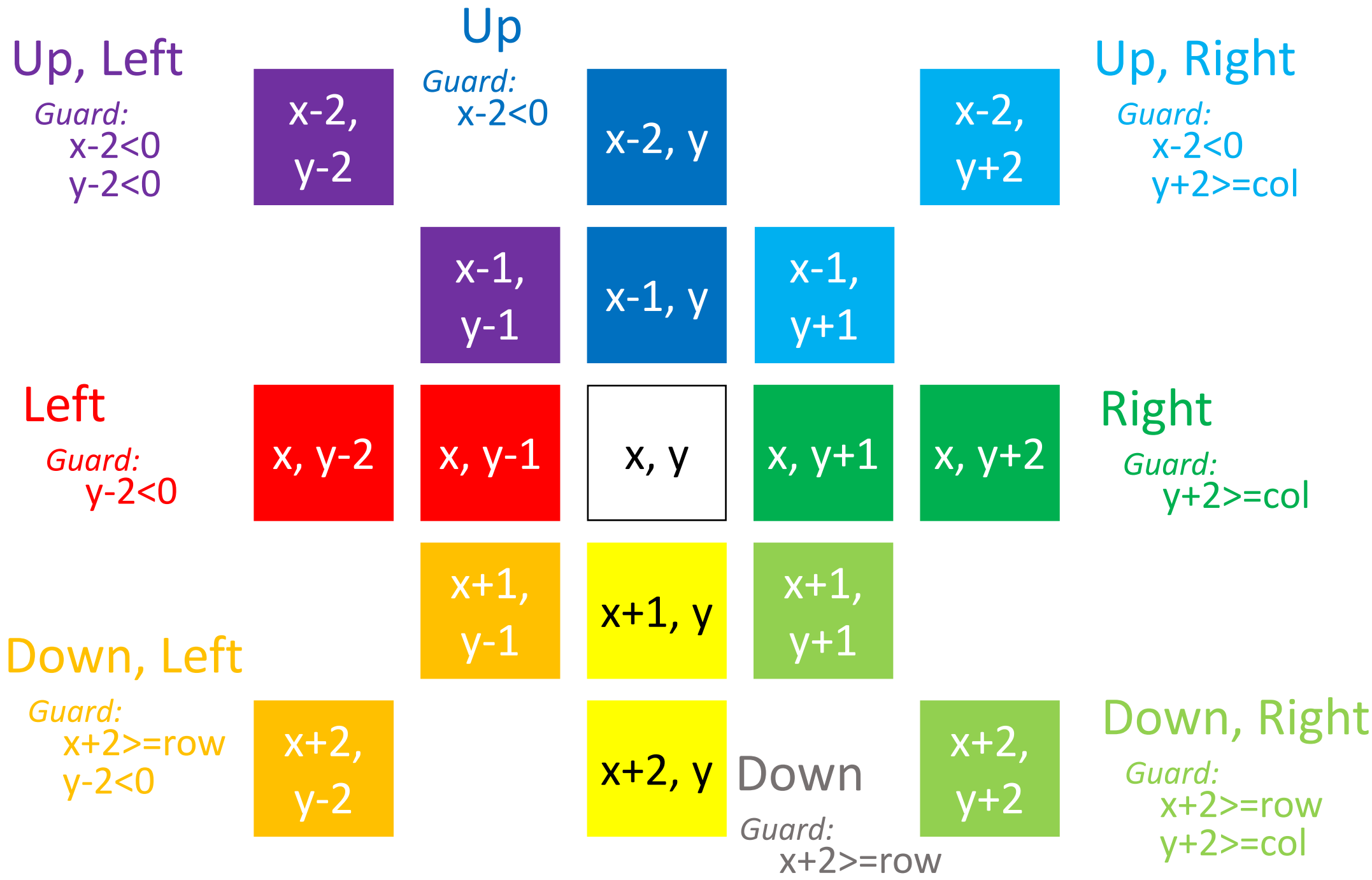
Move Down Right

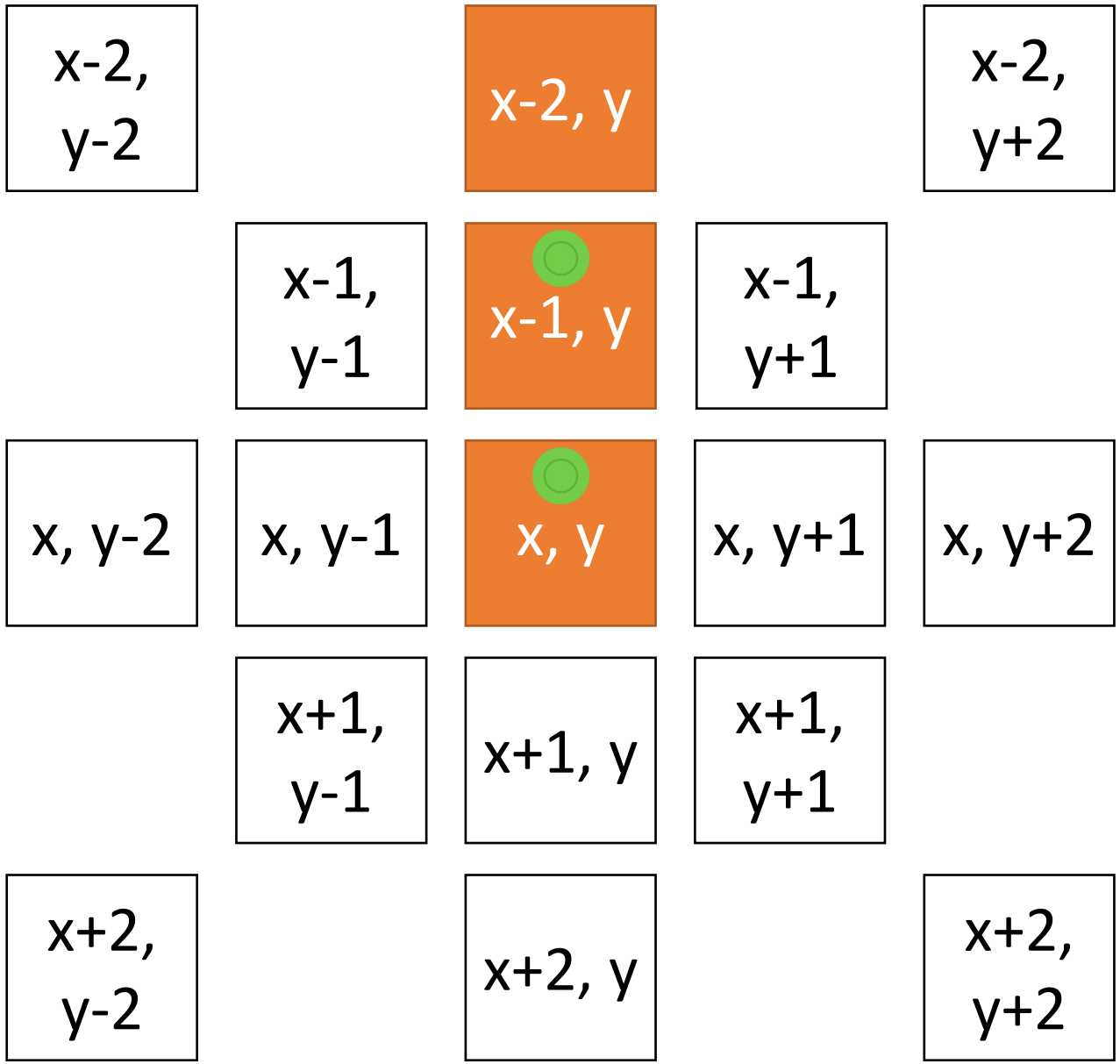
```
public void moveDownRight (int pos, int last)
{
      int startX = last / col;
      int startY = last % col;
      int endX = pos / col;
      int endY = pos % col;
      b [startX] [startY] = 0;
      b [startX + 1] [startY + 1] = 0;
      b [endX] [endY] = 1;
}
```



Calling Move Right Methods

```
//TO DO: Fill this comment in
else
{
    int n = Integer.parseInt (e.getActionCommand ());
    if (last == -1)
    {
        last = n;
    }
    else
    {
        if (isValidUp (n, last))
        {
            moveUp (n, last);
        }
        else if (isValidDownRight (n, last))
        {
            moveDownRight (n, last);
        }
        last = -1;
        //other checks go here
        redraw ();
    }

}
```


Up, Left
Guard:
x-2<0
y-2<0
x-2, y-2
Up
Guard:
x-2<0
x-2, y
x-2, y+2
Up, Right
Guard:
x-2<0
y+2>=col
x-1, y-1
x-1, y
x-1, y+1
Left
Guard:
y-2<0
x, y-2
x, y-1
x, y
x, y+1
x, y+2
Right
Guard:
y+2>=col
x+1, y-1
x+1, y
x+1, y+1
Down, Left
Guard:
x+2>=row
y-2<0
x+2, y-2
x+2, y
Down
Guard:
x+2>=row
x+2, y+2
Down, Right
Guard:
x+2>=row
y+2>=col


x-2, y-2
x-2, y
x-2, y+2
x-1, y-1
x-1, y
x-1, y+1
x, y-2
x, y-1
x, y
x, y+1
x, y+2
x+1, y-1
x+1, y
x+1, y+1
x+2, y-2
x+2, y
x+2, y+2