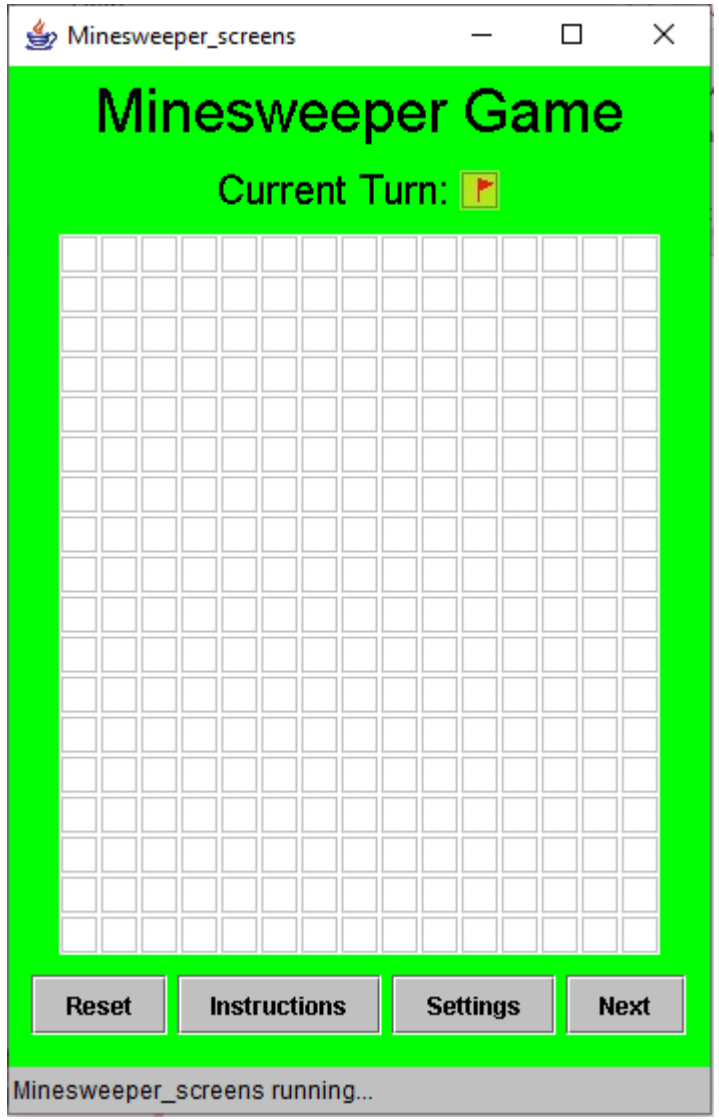
Minesweeper

Count Neighbours

```
//Game screen
JLabel turnPic;
int turn = 1;
int last = -1;
//grid
int row = 18;
int col = 15;
JButton a[] = new JButton [row * col];
int b[] [] = new int [row] [col];
int show[] [] = new int [row] [col];
int flags[] [] = new int [row] [col];

int levelCount = 10;

//Formatting
Color backgroundColour = Color.green;
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 (20, 20);
```



Re-write the Redraw

```
public void redraw ()
{
    int m = 0;
    for (int i = 0 ; i < row ; i++)
    {
        for (int j = 0 ; j < col ; j++)
        {
            if (show [i] [j] == 0 && flags [i] [j] == 0)
                a [m].setIcon (createImageIcon ("cover.jpg"));
            else if (show [i] [j] == 1 && flags [i] [j] == 0)
                a [m].setIcon (createImageIcon (b [i] [j] + ".jpg"));
            else if (show [i] [j] == 1 && flags [i] [j] == 1)
                a [m].setIcon (createImageIcon ("flag.jpg"));
            m++;
        }
    }
}
```

Add a Reveal

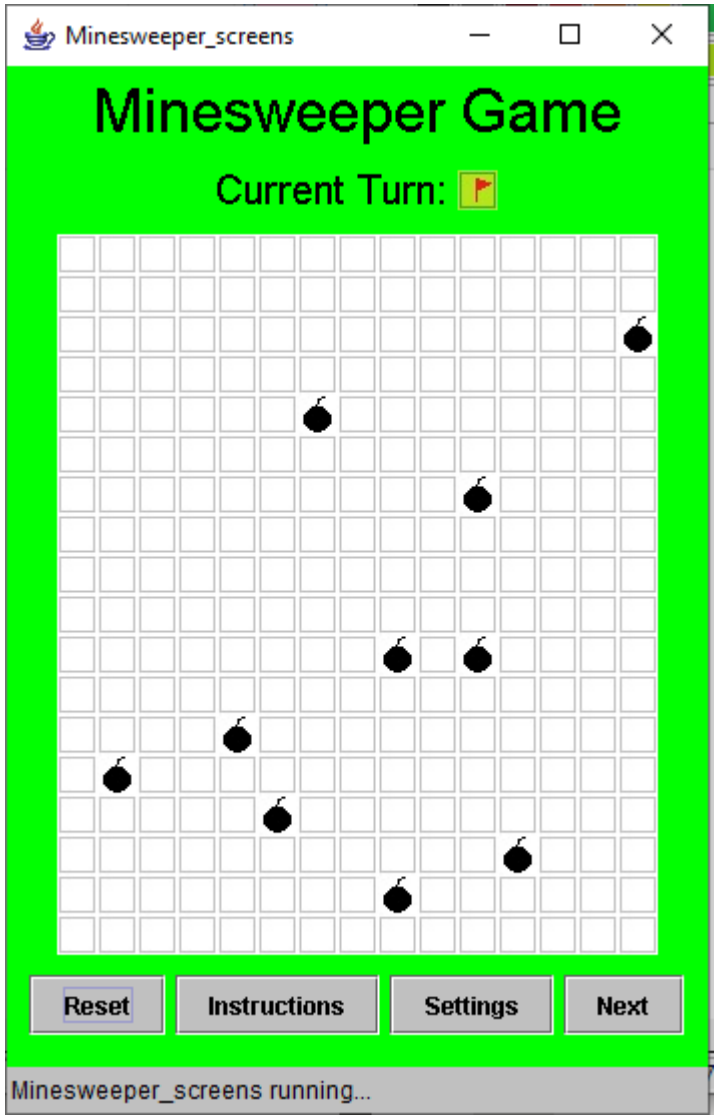
```
public void reveal ()
{
    int m = 0;
    for (int i = 0 ; i < row ; i++)
    {
        for (int j = 0 ; j < col ; j++)
        {
            a [m].setIcon (createImageIcon (b [i] [j] + ".jpg"));
            m++;
        }
    }
}
```

Add Mines

```
public void addMines (int amt)
{
    for (int i = 0 ; i < amt ; i++)
    {
        int x = (int) (Math.random () * row);
        int y = (int) (Math.random () * col);
        while (b [x] [y] != 0)
        {
            x = (int) (Math.random () * row);
            y = (int) (Math.random () * col);
        }
        b [x] [y] = 10;
    }
}
```

Temporary Add for Testing

```
//TO DO: Fill this comment in
else if (e.getActionCommand ().equals ("reset"))
{
    addMines (10);
    reveal ();
}
```



Counting Neighbours

```
public void neighbours ()
{
    for (int i = 0 ; i < row ; i++)
    {
        for (int j = 0 ; j < col ; j++)
        {
            if (b [i] [j] != 10)
            {
                int count = 0;
                if (i - 1 >= 0 && j - 1 >= 0 && b [i - 1] [j - 1] == 10)
                    count++;
                if (i - 1 >= 0 && b [i - 1] [j] == 10)
                    count++;
                if (i - 1 >= 0 && j + 1 < col && b [i - 1] [j + 1] == 10)
                    count++;
                //add other directions here

                b [i] [j] = count;
            }
        }
    }
}
```

Up, Left

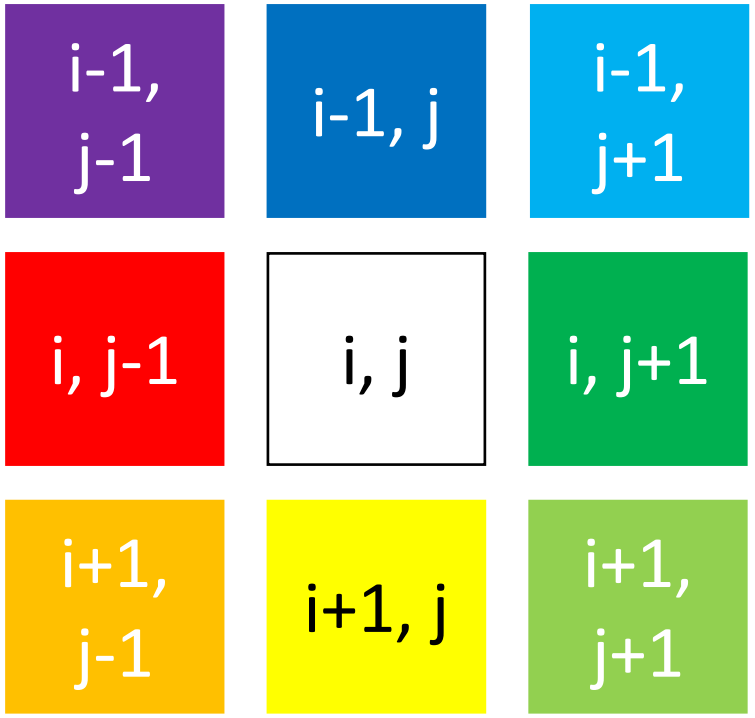
Guard:
i-1>=0
j-1>=0

Up

Guard:
i-1>=0

Up, Right

Guard:
i-1>=0
j+1<col



Left

Guard:
j-1>=0

Right

Guard:
j+1<col

```
public void neighbours ()
{
    for (int i = 0 ; i < row ; i++)
    {
        for (int j = 0 ; j < col ; j++)
        {
            if (b [i] [j] != 10)
            {
                int count = 0;
                if (i - 1 >= 0 && j - 1 >= 0 && b [i - 1] [j - 1] == 10)
                    count++;
                if (i - 1 >= 0 && b [i - 1] [j] == 10)
                    count++;
                if (i - 1 >= 0 && j + 1 < col && b [i - 1] [j + 1] == 10)
                    count++;
                //add other directions here

                b [i] [j] = count;
            }
        }
    }
}
```

Down, Left

Guard:
i+1<row
j-1>=0

Down

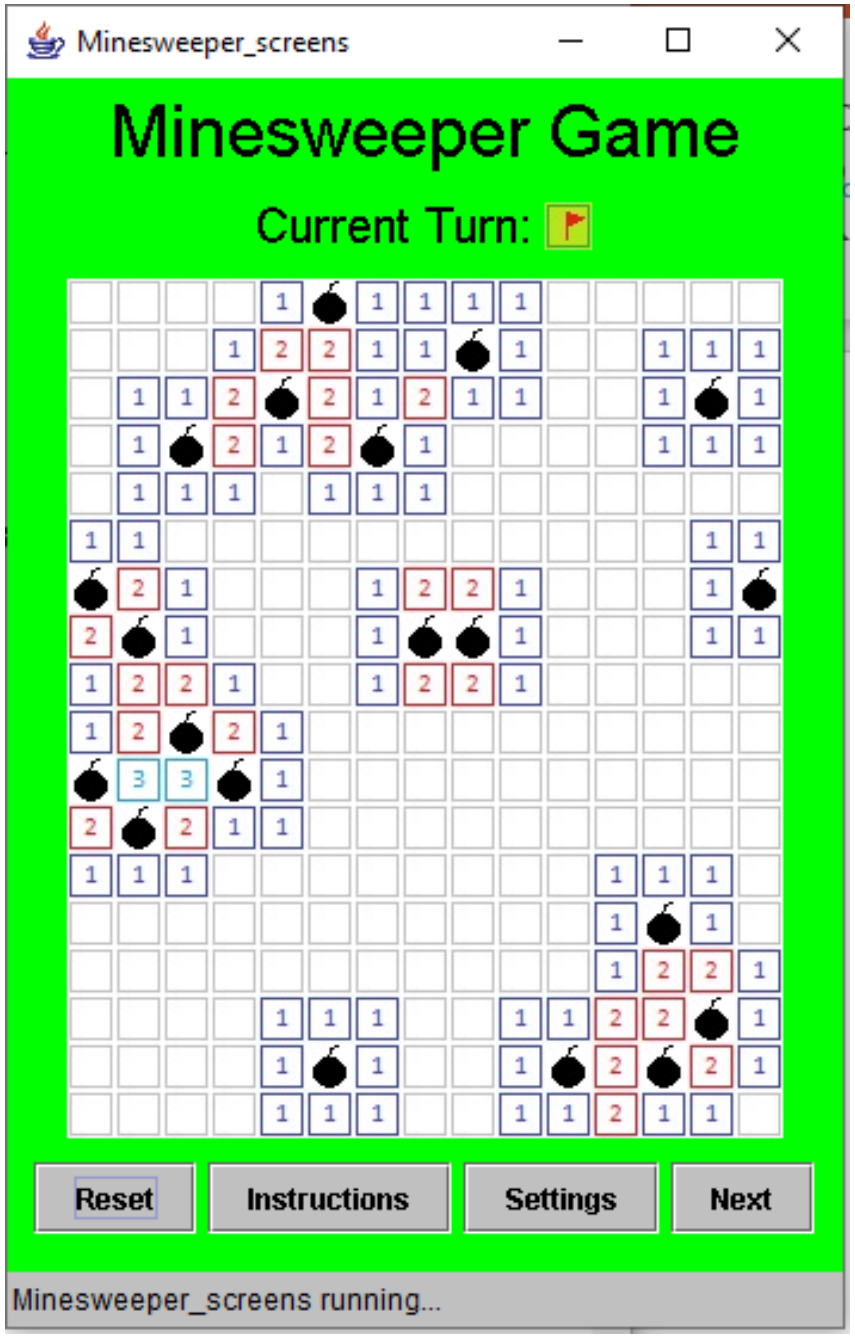
Guard:
i+1<row

Down, Right

Guard:
i+1<row
j+1<col

Update Temporary Add for Testing

```
//TO DO: Fill this comment in
else if (e.getActionCommand ().equals ("reset"))
{
    addMines (20);
    neighbours();
    reveal ();
}
```



Up, Left

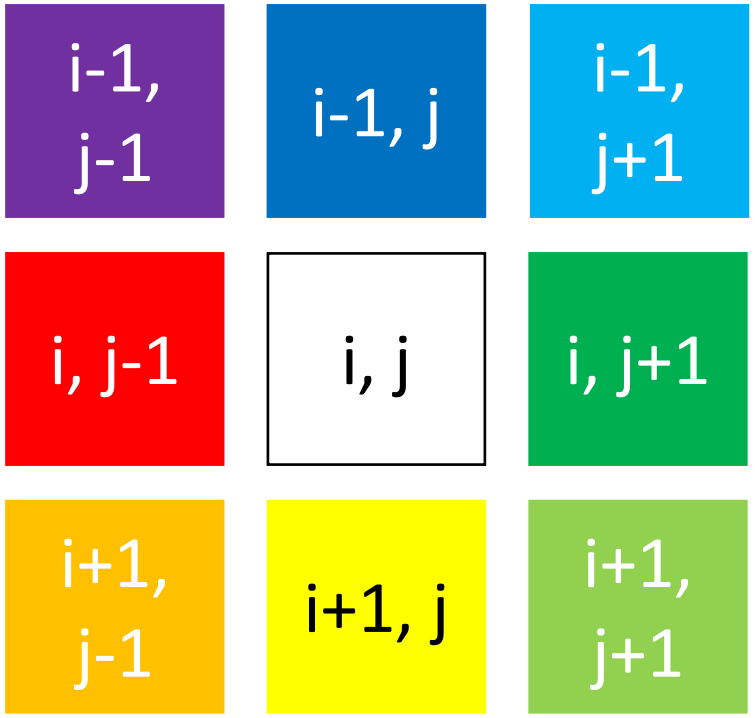
Guard:
i-1>=0
j-1>=0

Up

Guard:
i-1>=0

Up, Right

Guard:
i-1>=0
j+1<col



Left

Guard:
j-1>=0

Right

Guard:
j+1<col

Down, Left

Guard:
i+1<row
j-1>=0

Down

Guard:
i+1<row

Down, Right

Guard:
i+1<row
j+1<col

Open

```
public void open(int i, int j) {
   if (b[i][j] != 0)
      return;
   if (i-1 >= 0 && j-1 >= 0 && show[i-1][j-1] == 0){
      show[i - 1][j - 1] = 1;
      if (b[i - 1][j - 1] == 0)
         open(i - 1, j - 1);
   }
   if (i-1 >= 0 && show[i - 1][j] == 0) {
      show[i - 1][j] = 1;
      if (b[i - 1][j] == 0)
         open(i - 1, j);
   }
   //etc for all 8 neighbours
}
```