Here's the IMP.java source code

Ninety nine things to do:

1. Fix the reset function in the pulldown menu
2. Rotate image 90 degrees, odd shaped images should work
3. Show a histogram of the colors in a separate window
    1. Use the mapping function to normalize the distribution evenly
    2. https://en.wikipedia.org/wiki/Histogram_equalization
4. Turn a color image into a grayscale and display it.
5. Turn a color image into a grayscale image and then do a 3x3 mask to do an edge detection
6. Track a colored object.....orange is easiest. Results is a binary image that is black except where the colored object is located.

Use HIMP that I linked in above.

Grayscale

Use the Luminosity formula to turn a RGB image to a grayscale image that I discussed on **Jan. 20th.** ~~Done properly the you will need an array of bytes where you convert the original ints of each pixel into a byte of grayscale then you make that byte array a BufferedImage like this~~

~~BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRAY);~~

Once you get the grayscale value you can put it back into all three channels instead of making a byte array.

Then display bufferedImage back to HIMP.

Edge Detection

Start with a 3x3 mask (multiply each pixel and surrounding pixels by the multiplier) like this:

```
-1 -1 -1
-1  8 -1
-1 -1 -1
```

But if you want a much better line try this:

```
-1 -1 -1 -1 -1
-1  0  0  0 -1
-1  0 16  0 -1
-1  0  0  0 -1
-1 -1 -1 -1 -1
```

//This is in my histogram function in IMP

//first count all pixel values in R and G and B array

// Then pass those arrays to MyPanel constructor

//Then when button is pushed call drawHistogram in MyPanel.....you write DrawHistogram

//Don't forget to call repaint();

```
 JFrame redFrame = new JFrame("Red");
  redFrame.setSize(305, 600);
  redFrame.setLocation(800, 0);
  JFrame greenFrame = new JFrame("Green");
  greenFrame.setSize(305, 600);
  greenFrame.setLocation(1150, 0);
  JFrame blueFrame = new JFrame("blue");
  blueFrame.setSize(305, 600);
  blueFrame.setLocation(1450, 0);
  redPanel = new MyPanel(red);
  greenPanel = new MyPanel(green);
  bluePanel = new MyPanel(blue);
  redFrame.getContentPane().add(redPanel, BorderLayout.CENTER);
  redFrame.setVisible(true);
  greenFrame.getContentPane().add(greenPanel, BorderLayout.CENTER);
  greenFrame.setVisible(true);
  blueFrame.getContentPane().add(bluePanel, BorderLayout.CENTER);
  blueFrame.setVisible(true);
   start.setEnabled(true);
My panel class stuff that inherits from JPanel:
//instance fields
BufferedImage grid;
 Graphics2D gc;
///PaintComponent Method
 public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D)g;
        if(grid == null){
           int w = this.getWidth();
           int h = this.getHeight();
           grid = (BufferedImage)(this.createImage(w,h));
           gc = grid.createGraphics();

        }
        g2.drawImage(grid, null, 0, 0);

    }
```

Color Detection

Any pixels within a certain threshhold will be displayed white, non threshold pixels will be displayed black.