Program #4 - Design Notes

Program #4 Design notes… por vu.

1. Memorial Day 2015

I am **not** asking you to code over Memorial Day (what-a-guy).
But you **must** start thinking about P4 over this long weekend.

Write down on paper your design notes and questions. Sketch out some classes or UML.
On Wed May 25 @ 9:20am, I'll call you to the chalkboard (pressure, ha!) and ask you:

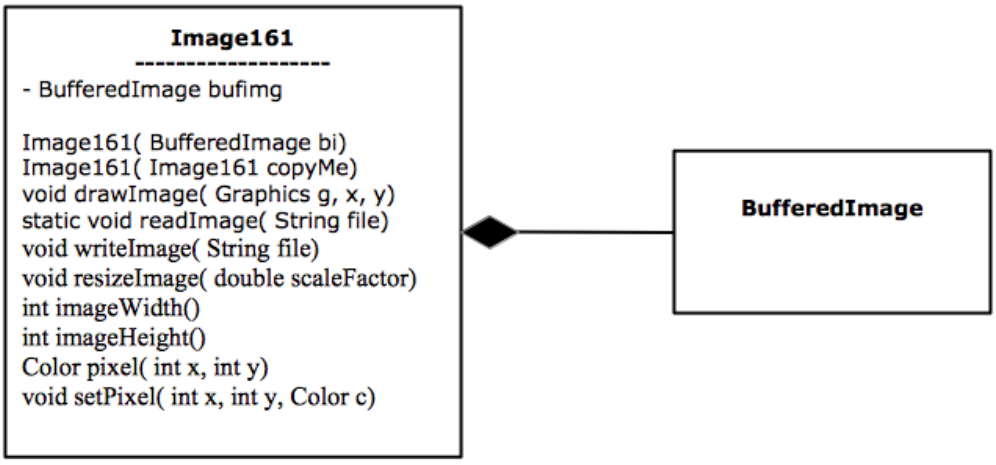
- What are your classes for P4?
- What design issues/questions do you currently have?
- What are the methods in your `ImageCollection` class?
- What are your first 5 baby steps to code up? Write it down!
- What are your 5 (or more) vacation pics? Are they on your k: drive?

It's OK to have questions. It's **not** OK to come empty-handed.

If you DO get a chance to code, then hit the easy stuff. Get your frame and panel up and running. Install a `KeyAdapter` and print the keys pressed in the panel. Try to draw an image in your panel with a hard-coded file name.

2. Image161

`Image161` is-a `BufferedImage`. I have tried to simplify image interaction: create, draw, read file, write file, resize, get/set pixels, etc. Can you see which methods will be useful in P4?
My code is in the k: drive. Here's some UML.



Notice how you draw an `Image161`… it requires a `Graphics` image. That means you'll probably draw it in the `paintComponent()` of your `JPanel`. This setup is similar to the `FunCircle` objects of Lab01 or `CloudString` objects in Program #2. Yes?

3. KeyAdapter

Look for KeyListener in the old Snippets.
Also, this official Java tutorial is good:
docs.oracle.com/javase/tutorial/uiswing/events/keylistener.html

From our design session Wed, regular chars and alternate chars (like arrow keys and escape) are handled differently. Here are two methods that you will override in your KeyListener.

```
// get a regular, old keyboard char press
@Override
public void keyTyped(KeyEvent e) {
    char ch = ke.getKeyChar();
    switch( ch) {
        case '?':
            // print help pane
            break;

        // more code here...

    }
}

// get a special/alt char like arrow keys or escape
@Override
public void keyPressed(KeyEvent e) {
    int code = ke.getKeyCode();
    switch( code) {
        case KeyEvent.VK_ESCAPE:
            // exit program
            break;

        // more code here...

    }
}
```

Another idea: make you code nice and clean by making one (private) method call in each case of your KeyListener methods: moveLeft(), moveUp(), makeTouristBigger(), etc.

4. Changing pixels, grayscale and more!

For one of my unique features in P4, I want to turn a color touring image into black and white, or grayscale. I really want that Mt. Rushmore shot!

The problem - I have a color image that I want to be black and white. What to do?

Well, there are zillions of algorithms. Here's an effective and easy algorithm for you.

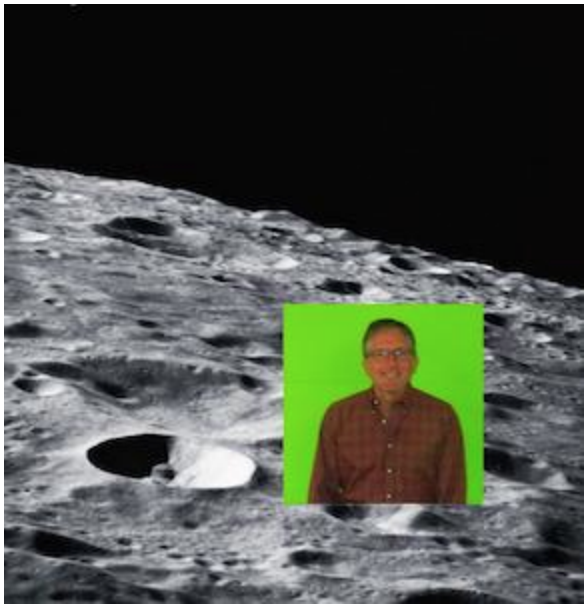
```
for each pixel {
  get pixel RGB
  ave = (red + blue + green) / 3
  set pixel RGB to (ave, ave, ave)
}
```

You can think of the average of RGB to be sort of the brightness of the pixel. The higher the average of RGB, the brighter the pixel will appear. So, Setting all RGB values to be the same, you get gray… and a higher average will lead to a brighter gray.

5. Green screen

Do green screen in two steps: 1) add one image (the tourist) to another (the vacation), and then 2) add images while ignoring green pixels.

In this pic, I add my image to the vacation image (the moon). Don't worry about the green yet.



The class `Image161` includes `getPixel()` and `setPixel()` methods, so you can a) get pixel values from the tourist and then b) set the pixel values of the vacation image.

```
// add img2 to the baseImage at location (x,y)
public static void addImage( Image161 baseImage,
            Image161 img2, int x, int y) { … }
```

Here's some pseudo-code for `addImage()`:

```
for x2 = 0 to img2 width {
   for y2 = 0 to img2 height {
      Color c = get img2 pixel at (x2, y2)
      set baseImage pixel at (x+x2, y+y2) = c
   }
}
```

Once you can add images, then add another method to do green screen addition.

```
// add green screen img2 to the baseImage at location (x,y)
public static void addGreenScreenImage( Image161 baseImage,
            Image161 img2, int x, int y, int greenBoost) { … }
```

The pseudo-code for this is the same except, inside the loop, where you add the pixels add this check for a green pixel:

```
if( (green + boost) > (red + blue)) {
   // set the pixel
}
```

This "boost" parameter give you a parameter to play with to make your green screen addition more or less sensitive to green in the tourist image.

6. ImageGallery

I'm going to try and work with you on `ImageGallery` in lab Wed because it's a good test case for using a `Map`. We'll see. If you have other things to work on before then, I would do that.

With William W's help, I think we have a very simple interface. Here are the public methods:

`ImageGallery()` - def ctor, the gallery is empty
`readGalleryFile( String fileName)` - read gallery file with name/image pairs
`String[] getImageNames()` - return an array of names of all gallery images
`Image161 readImage( String name)` - read and return the image with this name

So, your typical usage of `ImageGallery` will be 1) create one, 2) read its gallery file, 3) get an array of all image names, 4) choose one name and read it image.

7. Misc

I ran into a tiny problem when I changed my vacation image… my panel wouldn't resize. You need to `pack()` your frame for this to happen. But in my `VacationPanel`, I don't have a frame. If you run into this, here are some options:

- Don't worry about it - just manually resize your frame
- If that bugs you, then there is a (kind of) obscure way to get the frame from the panel. It's this snippet from www.stackoverflow.com/questions/9650874/java-swing-obtain-window-jframe-from-inside-a-jpanel .

```
JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(this);
frame.pack();
```

- If that weirdness bugs you too (my state of being), then you can move your `KeyListener` to your frame. When you do this, you'll have to have `VacationPanel` methods that do all the operations like `moveTourist()` and `resizeTourist()` and all. It's not hard, but it takes a little extra work.

Here's a simple way to show the user your vacation choices. It's another flavor of JOptionPane.

www.java2s.com/Tutorial/Java/0240__Swing/Todisplaysadialogwithalistofchoicesinadropdownlistbox.htm

So, my code has an array of vacation names (from my ImageGallery). When the user wants to change the vacation spot, then I show him the choices with code like this:

```
// code from my VacationPanel
String[] vacationSpots;
// vacationSpots = an array of names from my ImageGallery

String choice;
choice = (String) JOptionPane.showInputDialog( this,
                "Choose a virtual vacation spot", "Virtual Vacation",
                JOptionPane.QUESTION_MESSAGE, null, vacationSpots,
                vacationSpots[0]);
// choice is now the name in vacationSpots chosen by the user
```

Run faster!
Finish strong!
thanks… yow, bill

PS - early (fun) results… 161 on the moon!