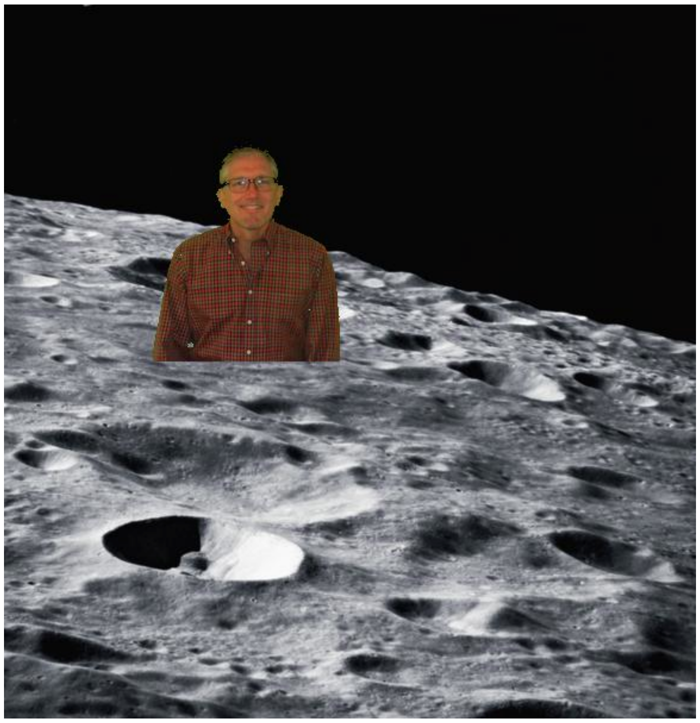
Program #4 - Virtual Vacation

“It’s even better than being there (TM)”

Logistics:

- Due: **Fri Jun 5, 2015**
- Worth: **10 points** (10% of your class grade)

Here’s a selfie I took on my virtual vacation to the moon. Pretty cool, eh? Eh!



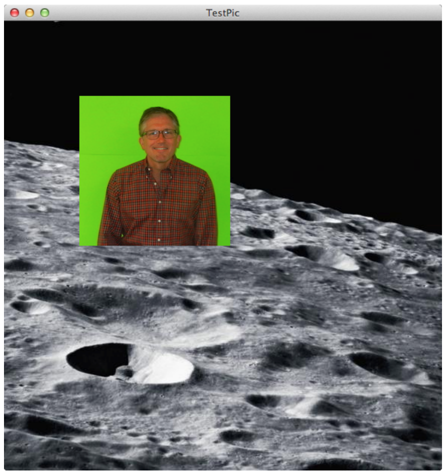
In Program #4, we’ll take some virtual vacations pic by placing ourselves (and loved ones) in locations we’ve may never have been (like the moon).

Program #4 is fun, and we’ll use:

- `JFileChooser` - to choose a tourist image file
- `KeyAdapter` - to control your gui using the keyboard
- `JPanel` - overriding `paintComponent()`
- `BufferedImage` - standard Java image class
- Image processing - diddling with the pixels to change an image
- Java - copy ctor and static methods, `ArrayList` (of course)

1. Description

OK, here's the basic idea. I start with a vacation image (the moon) and a tourist image (Prof Bill). The tourist image has a green screen background.



I move the tourist image in place with the arrow keys (up, up, right, down, etc). I make the image a little smaller with the minus sign (-). Once I find the spot and size I like, then I hit "g" for "green screen", and my tourist image is fixed in place and added to my vacation image… except for the green part. Huzzah!

Details!

a. Two images

At any point, you'll be dealing with two images in your panel:

- vacation image - the background image of some exclusive vacation spot
- tourist image - the foreground image of you or some other virtual tourist

b. Vacation image collection

One input to Program #4 is a file name. It's a CSV file of 1) vacation image names and 2) file names. The CSV file looks like this:

```
# vacation images for Program #4
Mt. Rushmore,vacation/rushmore.jpg
Chicago Bean,vacation/chicago.jpg
The Moon,vacation/moon.jpg
Noctrl Carnegie,vacation/noctrl_carnegie.jpg
```

Program #4 lets the user choose his/her vacation image from a list by name.

c. Green screen tourist
Show the user a `JFileChooser` to select a tourist image.

d. Keyboard control
Let's be clever and drive our GUI program using keyboard commands (`KeyAdapter` class!). Your minimal command set is:

Key(s)	Operation
v	Vacation image select (list)
t	Tourist image select (file chooser)
Arrow keys	move tourist image around
+	increase tourist image size
-	decrease tourist image size
g	Green screen merge
s	Save image to a file
?	Help message
Escape key	Exit

Own it!
You must add at least 3 more commands/features unique to your Program #4 implementation. A couple ideas:

- Create an "o" command that returns your tourist image to its Original state
- Create an "m" command that shows a menu of commands
- Create a splash screen for your program using an image you created (check out text for "splash screens")
- Make the tourist image grayscale, or some other change

Also, please contribute at least 5 vacation images to share on the k: drive. My five vacation images are already there.
Last program, so you know this… Own it to win, baby!

3. Design

Image161 - First step: grab my `Image161` class from the k: drive. It's a kinder and gentler Java image representation, and I have coded up some of the gobbledygook you need to read and resize images. You will add some methods here as well.

ImageCollection - Please create a class to manage a collection of image files: your vacation images. I called mine `ImageCollection`. It should read the CSV file that defines your vacation spots. We'll work on the design of this class in, um, class.

is-a KeyAdapter - I have hints to help you with your `KeyAdapter`, and other blather in our Design Notes. There's already an old Snippet on KeyListener, if you're interested.

We'll have some class time on this design, and some design notes as well (see below).

4. Grading

Create a `program4` folder in your k: drive.
Place these files in that folder:

- A `README` file describing the state of your program
- All the Java files that comprise your Program #4 solution
- And of course… show me your **favorite** virtual vacation images!!!!

All your code must follow our 161 Coding Guidelines. Ugly code will be penalized with a 0-100% reduction in points. A program that doesn't even compile is worth 0 points.

Finish strong on our last program.
yow, bill

PS - Design note goodness:

Program #4 Design Notes