***The* University *of* Western Ontario**
CS1026a - Summer 2015
Assignment 2

DUE: Thursday, June 4th, 11:55pm
Weight: 10%

Purpose:
To gain experience with:

- Using methods
- Using loops
- Picture manipulation

Part A:

You are to create a total of **6 object methods** within the *Picture* class. These methods will perform some kind of picture modification (ex. Grayscale, mirror, flip, make more red, blur, etc.). You can check out lecture slides, lecture code, labs, or the textbook for examples. For these methods, **4** need to be original, and **2** can be from other resources (see examples above). Be creative! Have fun!

Functional Specifications for part A:

1. Add the 6 methods to the *Picture* class (the *Picture.java* file) after the commented line:

```
////////////////// methods ////////////////////
```

2. You will call these newly created methods from your own class called *Assign2*. Create a file called *Assign2.java* and add the following code:

```
import java.awt.Color;
public class Assign2
{
     public static void main(String[] args)
     {
          /* Insert your code here */
     }
}
```

You will need to create the picture objects and invoke your object methods by adding the appropriate code within the main method. See lecture code for hints on how to do this. **Remember**, you will need to show() the picture after you have made the changes in order to see what has actually changed.

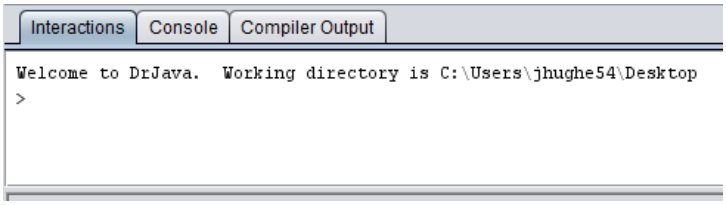
A high level snapshot of the algorithm will look something line this:

- Create a new picture object and show it on the screen
- For each of the 6 methods:
- Create a new image that's a copy of the original
    - Change the image with your method
    - Show the changed image to the screen

You can use whatever – appropriate – .jpg you want: you can take a selfie, you can use a picture you have taken, or you can use a picture provided by the textbook. **However**, it would be ideal if the picture had a lowish resolution for part B.

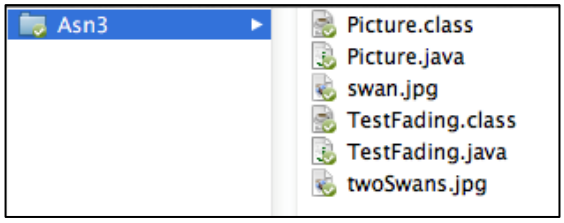
Note: The working directory – Use files *without* FileChooser

The working directory is what programmers call the folder (directory) where the program is being run. In Dr. Java, the working directory is the same as the location of the ".class" file that you are running. You can see the working directory listed at the top of the interactions pane when you hit **"Run"** in Dr. Java. In the figure below, my working directory is "C:\jhughe54\Desktop".



To use files that are saved in the working directory in your program, you simply list their name and the program will know to look in the working directory for that file. If it is not there, you will get an error when you try to load the Picture file.

For example, if I had the files for this Part A in the folder "Asn3", along with two images like this:



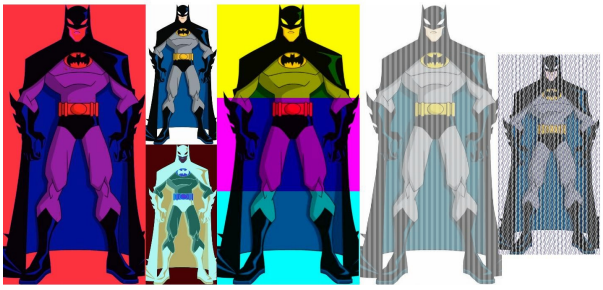
I could create a picture object using the following line since swan.jpg is within the working directory.

```
Picture startPic = new Picture("swan.jpg");
```

You **must** use a file from the working directory for this assignment. You will submit the picture files you want us to use to OWL with your .java files. **You cannot use a file chooser!**

Part B:

Make a collage of modified pictures! In this part you are to make a large picture consisting of modified versions of your original picture. Your collage must consist of at least 6 modified versions of your original image (6 versions modified with the object methods written for **Part A**). Feel free to include more method and more than 6 modified versions of the original image in the collage. Your collage may contain the original image; however, if you choose to include it, then you must have at least 7 versions of the original image (6 new + 1 original). See high quality examples below:



1. You may arrange the images however you would like.
2. The original file must be loaded from the working directory (see above).
3. Remove the code from the main method from Part A and replace it with the code required to do Part B
4. You **must** also save the resulting collage with the file name *"myCollage.jpg"* (without quotes).

```
/*assume we have a picture object called myPic */
myPic.write("MyCollage.jpg");
```

Resizing:

Since your original image can be one of the ones provided or your own, you may need to reduce the size of the image in order to use it properly. This can be done using commercial software or the method `getPictureWithHeight` in the picture class. This method will create a picture with the specified height from an existing picture, and will keep the same height/width ratio. Here is an example of how to use the method:

```
/*assume you are starting with a Picture object
      referenced by a reference variable called largerPic */

   Picture smallerPic = largerPic.getPictureWithHeight(200);
   smallerPic.write("filepath goes here /smallerPic.jpg");
```

Non-functional Specifications:

1. Include comments for your classes

```
// CS1026B Assignment 2
// Your name
// A brief description of what this program does
```

2. Include brief comments in your code that explain what each method is doing.
3. Include comments for any code which is unclear.
4. Assignments are to be done individually and must be your own work. Software will be used to detect cheating.
5. Use Java conventions and good Java programming techniques, for example:
   i. Meaningful variable names
   ii. Conventions for naming variables and constants
   iii. Use of constants where appropriate
   iv. Readability: indentation, white space, consistency

Important! You must follow all the specifications above for your program

What You Will Be Marked On:

1. Functional specifications:
   - ✓ Are the required methods written according to specifications?
   - ✓ Do they work as specified? Pay attention to details!
   - ✓ Are they called as specified?
   - ✓ Are the two objects drawn according to the specifications?
2. Non-functional specifications: as described above

What to Submit:

*Submit .java documents. Be VERY careful **not** to submit the .java~ version or the .class file. This can be tricky, as your system may not show the file extension in the file selection dialog.*

1. *Picture.java* (with your **new methods**)
2. *Assign2.java* (with the proper code to run Part B)
3. The JPEG file with the original image used for creating your collage
4. The file *myCollage.jpg*

How to Submit:

You are to submit the specified files through the online OWL submission for Assignment 2 under the "Assignments" tab.