ICS114 Lab 8

GOALS:

a. to download the textbook mediafiles
b. to use the Picture and Pixel classes
c. to learn how to create jar files
d. to gain practice with arrays

1. Download and unzip the files:

a. `mediasources-no-movies.zip`
This file contains digital files such as pictures.

b. `bookClasses.zip`
This file contains the most recent copy of the code from the textbook. We will use this instead of `ICS114 Code` from now on.

2. Download: `textbook1-1.jar.` This contains the most recent textbook code in a jar file format. You should put this in a folder where it is easily accessible. For example, if you have a 'Labs' folder that contains a folder for each of our labs, put the jar file there.

3. Download: `FileChooserDemo.java.` Compile and run it with the new jar file. Recall to compile with a jar file, you must specify a 'classpath.' If your jar file is in your Labs

folder and you are in your Lab8 folder, to compile, use the command:

```
javac –cp ..\textbook1-1.jar;. FileChooserDemo.java
```

and to run:

```
java –cp ..\textbook1-1.jar;. FileChooserDemo
```

When asked to select a file, select the file `caterpillar.jpg` which is in the directory: `mediasources-no-movies\intro-prog-java\mediasources`

Observe how you can move around the picture, adjust the zoom, see the pixel coordinates and their colours.

Examine the source code of `FileChooserDemo.java`.

a. How many objects are there?

b. What object methods are being called? To what classes to those belong?

c. What class methods are being called? To what classes to those belong?

4. Add a method to the Picture class to decrease the red in a picture.

a. Insert the following incomplete method into the picture class . Complete it as indicated by the comments. Observe how we can loop through all of

the objects in an array.

```
/* Complete this method as indicated in the comments
   and put it in the Picture class.
*/
/*  0. This method belongs in the picture class.

    1. Complete the header. The method is called decreaseRed.
       It does not return anything and has no parameters.
*/
public
  {
    /* 2. Declare a reference variable for an array of pixels.
    */

    /* 3. Use the method getPixels() from the Pixel class to
          get the pixel array of the current object.
    */

    /* 4. Declare an integer variable.
    */

    // 5. loop through all the pixels in the array
    for (Pixel pixelObj : pixelArray)
    {
        // 6. get the red value of the current pixel and store
        //    it in the integer variable you declared above.

        // 7. decrease that red value by 50%

        // 8. set the red value of the current pixel to the new, reduced

    }
  }
```

b. Compile the picture class.

c. Now, we must create a new jar file in order to see the changes in the picture class. Create a jar file with:
`jar cvf textbook1-1.jar *.*`
Next, copy this new file into your labs file, overwriting the original jar file you downloaded earlier.

d. Download the file `ManipulatePixels.java`. Compile and run this file to test your newly written method, `decreaseRed`.

5. Look at the 'javadoc' documentation that comes with the textbook classes. Specifically, look at the Pixel class' object methods. Write and add another method to the Picture class that manipulates pixels in a way of your choice. Use at least one conditional statement. Be sure to test your method by calling it from the ManipulatePixels program.

6. Practice with integer arrays. Complete the following array exercises (from the lecture notes). Create a Java class called ArrayExercises. These methods will be static methods, as we will not require an object to invoke them. The methods will be invoked from the main method indicated in 8. The main method will create and initialize

an integer array which will be passed into the methods.

Array Exercises

1. Write a method that displays the contents of an array to the screen, one element per line. Formal parameter: an integer array.

2. Write a method that outputs the contents of an array in reverse order. Formal parameter: an integer array.

3. Write a method that returns the maximum value in an array. Formal parameter: an integer array.

4. Write a method that rearranges the contents of an array into reverse order. (i.e. actually change the order in which the elements are stored in the array. Formal parameter: an integer array.

5. Write a method that returns the index where a particular element is found in an array. Formal parameters: an integer array, an integer.

6. Write a method that returns true if a particular element is found in an array. Formal parameters: an integer array, an integer.

7. Write a method that returns the number of times a particular element occurs in an array.

8. Write a main method to test each of the methods you wrote. Present the user with a menu allowing them to chose which method they wish to run.