

```

import java.util.Scanner;
import java.io.*;
//import com.sun.image.codec.jpeg.*;
import java.awt.*;
import java.awt.image.*;
import org.cmc.sanselan.sampleUsage.*;

public class JPEGStripper{
    public static void main(String[] args) throws Exception{
        Scanner reader = new Scanner(System.in);
        int numFiles = 0; //storage
        point for number of files. Will be used to control the number of times for loops are run
        boolean TASK_metadata = false; //boolean
        variables for determining which processing techniques will be run.
        boolean TASK_quality = false; //Because
        they are set to false by default, I only need to check later if they are true
        boolean TASK_scale = false;
        File dir = new File("."); //Gets the
        current directory so that files can be filtered.

        /* Set up the filter to take on .jpg files (this is a variable, for some reason) */
        FilenameFilter filter = new FilenameFilter() {
            public boolean accept(File dir, String name) {
                return name.endsWith(".jpg") && !name.startsWith(".");
            }
        };

        /* Intro */
        System.out.println("\n.: JPEG Stripper, by Gordon Diggs - 2008 :.");
        System.out.println(".: Using Code from Marco Schmidt (much thanks) :.");
        System.out.println(".: and the Senselan library (because Imagero sucks) :.\n\n");

        /* Get Files from Current Folder */
        String[] victims = dir.list(filter);
        numFiles = victims.length;

        /* How Many Files? */
        System.out.println("Found " + numFiles + " images.");

        /* Make Directory 'originals' */
        /* This uses the Process data type to run an external command */
        Process mkdirCMD = Runtime.getRuntime().exec("java CreateDirectory");
        mkdirCMD.waitFor();
        System.out.println("\nDirectory 'originals' Created\n");

        /* Copy Files to folder */
        /* This uses Marco Schmidt's CopyFile class and an external Process command like the one above */
        /* This does not work with files with spaces in them, but that is a non-issue, seeing as spaces are
        never used in web filenames. */
        for(int i=0; i<numFiles; i++){
            System.out.println("Copying \" + victims[i] + "\"");
            Process copyFilecurr = Runtime.getRuntime().exec("java CopyFile " + victims[i] + "
originals");

```

```
        copyFilecurr.waitFor();
    }
    System.out.print("\n");

    /* If no Images, quit the program */
    if(numFiles == 0){
        System.out.println("No Files Found, exiting program.\n");
        System.exit(0);
    }

    /* Metadata Choice */
    System.out.print("Would you like to strip metadata from the images? (Y/N) ");
    String temp1 = reader.next();
    if(temp1.equalsIgnoreCase("y")){
        TASK_metadata = true;
        temp1 = null;
    }

    /* Scale Choice */
    System.out.print("Would you like to scale the image? (Y/N) ");
    String temp3 = reader.next();
    if(temp3.equalsIgnoreCase("y")){
        TASK_scale = true;
        temp3 = null;
    }

    /* Quality Choice */
    System.out.print("Would you like to change the image quality? (Y/N) ");
    String temp2 = reader.next();
    if(temp2.equalsIgnoreCase("y")){
        TASK_quality = true;
        temp2 = null;
    }

    /* Perform Metadata Stripping */
    if(TASK_metadata == true){
        System.out.println("\n:: Stripping Metadata ::");
        for(int i=0; i<victims.length; i++){
            System.out.println("Stripping " + victims[i]);
            StripIMG(victims[i]);
        }
    }

    /* Perform Scaling */
    if(TASK_scale == true){
        System.out.println("\n:: Scaling Images ::");
        System.out.print("What should the max size (in pixels) of an image side be? ");
        int imgSIZE = reader.nextInt();
        for(int i=0; i<numFiles; i++){
            System.out.println("Scaling " + victims[i]);
            ScaleIMG(victims[i], imgSIZE, 100);
        }
    }
}
```

```

/* Perform Quality Shift */
if (TASK_quality == true) {
    System.out.print("\nWhat should the quality be? (0-100) ");
    int imgQUAL = reader.nextInt();
    for (int i=0; i<numFiles; i++) {
        System.out.println("Shifting " + victims[i]);
        ShiftIMG(victims[i], imgQUAL);
    }
}

System.out.println("\nAll Done!\n");

/* DEBUG!!!! */
//System.out.println("\n::WATCH ME DEBUG\n::Number of Images: " + numFiles + "\n::Metadata: " + TASK_metadata + "\n::Quality: " +
TASK_quality + "\n::Scale: " + TASK_scale + "\n");
}

private static void StripIMG(String JPEGFILE) throws Exception{
    File jpg = new File(JPEGFILE); //creates
the file object of the original
    File jpgnew = new File("temp.jpg"); //then the
temporary file (Senselan cannot do this with only one file)

    WriteExifMetadataExample exifGOD = new WriteExifMetadataExample();

    exifGOD.removeExifMetadata(jpg, jpgnew); //my
favorite method in the entire world

    jpg.delete(); //delete the
original (still has metadata)
    jpgnew.renameTo(jpg); //rename
the new one (no metadata) to the name of the original. delete and renameTo are methods of the File class
}

private static void ScaleIMG(String JPEGFILE, int size, int quality) throws Exception{
    Process scaling = Runtime.getRuntime().exec("java Thumbnail " + JPEGFILE + " " +
JPEGFILE + " " + size + " " + size + " " + quality);
    scaling.waitFor();
}

private static void ShiftIMG(String JPEGFILE, int quality) throws Exception{
    Process shifting = Runtime.getRuntime().exec("java Quality " + JPEGFILE + " " + JPEGFILE
+ " " + quality);
    shifting.waitFor();
}
}

```