Sarajevo School of Science and Technology
CS/IS 160 Programming and Problem Solving II
May 28, 2015

Week 13 Lab Exercises
Exceptions, File and Image I/O

In this lab, we will explore how to handle exceptions while working with file and image I/O.

Lab setup:

- Create a project named Week-13-Lastname-Firstname
- Use the *default* package or create appropriately named packages for your classes

1. Reverse lines in a text file (50 points).

Implement a **main** method to open an input text file so that you can read its lines one at a time. Read each line as a **String** and place it into an **ArrayList**. Then open an output text file so that you can write text into the file line by line. Print all of the lines in the **ArrayList** to the output file in reverse order. Therefore, the first line in the input text file will be the last line in the output text file and vice versa. You can use any example text file as your input file, and you can name the output file as **reversed.txt**.

Hints: To reverse an array list, you can iterate over it backwards or you can use `reverse` static method from `java.util.Collections`. One way to write to an output text file is using the `java.io.PrintWriter` class. You can use the following code to read a text file line by line.

```
// open a file reader and wrap it in a buffer
FileReader fr = new FileReader("file_name_here.txt");
BufferedReader br = new BufferedReader(fr);

// read line by line and add it to the lines array
String line = br.readLine();
while( line != null ) {

      // TODO: add each line to your array list here

      line = br.readLine();
}
```

2. Black & white filtering of an image (50 points).

Write a program that will use static methods from the `javax.imageio.ImageIO` class to read and write an image file. After successfully reading an image of your choice (download one from the web and place it in the project directory), call the `applyBlackWhiteFilter` method on the image, and then write it out to a new output image file. You can use the following code to perform these actions, but you will need to handle possible IO exceptions.

```
   BufferedImage image = ImageIO.read(new File("sarajevo.jpg"));
   applyBlackWhiteFilter(image);
   ImageIO.write(image, "jpeg", new File("sarajevo-BW.jpg"));
```

The following method takes a colour image and turns it into a black-and-white one, by setting the red, green, and blue value to their average for each image pixel. Experiment with this method and see what happens to the image. For example, set each colour value to one of the component values (say the red one) for each pixel. What do you see? Does it make sense?

```
    /**
     * Apply black and white filter to a buffered (in-memory) image.
     * @param img Image to modify
     */
    public static void applyBlackWhiteFilter(BufferedImage img) {

        int width = img.getWidth();
        int height = img.getHeight();

        WritableRaster raster = img.getRaster();

        for(int i=0; i < width; i++) {
            for(int j=0; j < height; j++) {

                int[] pixelRGB = new int[3];
                raster.getPixel(i, j, pixelRGB);

                int r = pixelRGB[0];
                int g = pixelRGB[1];
                int b = pixelRGB[2];

                int gry = (r + g + b) / 3;
                pixelRGB[0] = gry;
                pixelRGB[1] = gry;
                pixelRGB[2] = gry;

                raster.setPixel(i, j, pixelRGB);
            }
        }
    }
```