

## AP COMPUTER SCIENCE A – HOMEWORK #6

Steganography is the practice of hiding information in images. As we have discussed before, the color of each pixel in an image is represented by an integer. The four bytes of the integer (`BufferedImage.TYPE_INT_ARGB`) contains the alpha, red, green and blue components of the pixel's color, with each byte ranging from `0x00 = 0` to `0xff = 255`. For red, green and blue, the value 0 corresponds to no intensity; 255 to full intensity. For alpha, the value 0 corresponds to full transparency; 255 to full opacity. In any case, this situation is depicted below.



Note that, in the above figure, the two least significant bits of each byte are shaded. If these eight (shaded) bits were all to be set to zero, then the color of the pixel would be changed slightly (if not imperceptibly). Similarly, if one byte of information is placed into these eight (shaded) bits, then again, the pixel's color would be changed only slightly. Thus, one byte of information can be hidden in the pixel's integer, and later, this information can be retrieved.

The goal of this assignment is to write a program `ImageMessage.java` which hides information (specifically, a message) in an image using the technique described above. The program will be able to read in an image, embed a message in it, and the output the (slightly corrupted) image. The program will also be able to read in an image with a message so-embedded, and then retrieve the message.

Your program will use the class `ImageRW` as is. This class has two static methods. One is `public static BufferedImage readImage ( String fileName )`, which reads an image file into your program. The other is `public static void writeImage ( BufferedImage bi, String fileName )`, which outputs the image `bi`.

Your program will also use class `PixelOps` as is. This class also has two static methods. The method `public static int embed ( int pixel, int byteToEmbed )` embeds the byte into the pixel integer, and returns the (slightly corrupted) pixel. The method `public static int retrieveByte ( int pixel )` retrieves the embedded byte of information from the (slightly corrupted) pixel integer.

Finally, your program will also use the class `Keyboard` as is. You used this class earlier in your Caesar cipher program.

- 1) Add a method `public boolean equals ( IntPair o )` to `IntPairStub.java` to make a class `IntPair`. The `equals` method should return whether or not this `IntPair` is equal to `IntPair o`. This class will be used as a container for the  $x$  and  $y$  coordinates of a pixel in an image.
- 2) Starting with `RandomPixelListStub.java`, follow the directions in the comments to add the necessary code to the constructor to make a class `RandomPixelList`. Note that the field `rpl` in this class contains the list of pixels into which the message will be embedded.

## AP COMPUTER SCIENCE A – HOMEWORK #6

- 3) Starting from `ImageMessageStub.java`, create a class `ImageMessage` by:
- a) Adding code to method `private static void embedMessage`, by following the directions in the comments, to embed the message in the image.
  - b) Adding code to method `private static void retrieveMessage` to retrieve, and print to the console, the message embedded in the image.
  - c) In `main`, where indicated, use a `ConsoleInput` object to read in the option (1 = embed message, 2 = retrieve message). Use a `while` loop and re-prompt the user until a valid option is entered.
  - d) In `main`, where indicated, use the `ConsoleInput` object to read in the password and input file name. If option 1 has been chosen, also read in the output file name and the message to embed. Use a `while` loop to re-prompt the user if the message is longer than `MAX_MESSAGE_LENGTH`. Your code should print out something like `The maximum message length is 255` if the message is too long (and re-prompt the user).
  - e) You should ensure that the input image is large enough to accommodate `MAX_MESSAGE_LENGTH+1` embedded bytes. If not, print out an error message and `return`.
  - f) Finally, call either method `embedMessage` or `retrieveMessage` as appropriate according to the option chosen.
- 4) Try running your code to extract a message from an image that does not contain a message, or using the wrong password to extract a message from an image which does contain one. In each case, your code will bomb out. Fix your code not to bomb out in such a case, and instead, print out something like `"scarlett.png" does not contain a message` to the console.

Once you are convinced that your program is working correctly, email the files `IntPair.java`, `RandomPixelList.java` and `ImageMessage.java` to me at [sharren@d131.org](mailto:sharren@d131.org).