Name: _____

Date: _____

ACTIVITY 4

Hiding and Revealing a Text Message

Information can be represented in various ways, as seen throughout this lab with images being stored as 2D arrays of pixels. All information stored on a computer is stored as sequences of bits, and how those bits are interpreted can completely change their meaning. Consider the following eight-bit sequence—0100 1101. It's impossible to know what it represents without context. It could be the integer 77, the character 'N', the amount of red in a pixel, or any number of things.

The biggest consideration when determining how to represent information is knowing how many items need to be represented, since this will ultimately determine the number of bits that will be used. For example, early monitors were not capable of displaying the variance in color that current monitors can display, so colors were represented using far fewer bits. As monitors have improved and become capable of displaying more colors, the number of bits used to store color has increased so that all possible colors that can be displayed have a unique representation.

1. How many items can be represented with the following numbers of bits:

2 bits? 4 4 bits? 16 8 bits? 256

This activity involves hiding and then revealing a text message in a picture. The ideas are the same as those involved in hiding and revealing a picture, but the text message must be split up and reassembled more carefully.

Recall that to hide a picture, the leftmost two bits in each color for each pixel from the secret image are stored in the source image as the rightmost two bits in each matching color/pixel. The picture can be revealed by taking those rightmost bits and making them the leftmost bits again.

In this activity, you're going to store messages consisting of uppercase letters and spaces in a picture. To do this, the 26 letters will be represented by numbers (1–26 where A = 1, etc.), a space will be represented using the value 27, and a 0 will represent the end of the string. In this way, messages will be able to be coded using 28 distinct integer values.

By encoding a message in this way, five bits will be needed to hold each letter or space. Recall that five bits can hold the decimal values ranging from 0 to 31 so there are even a few extra bits in case punctuation is desired. The desire is for the text to be truly hidden in the picture, so only the rightmost two bits in each color value of each pixel are used to store the coded message. Since there are three color components per pixel, each coded character will be split into three pairs of two bits where the leftmost bit is always 0. Again, more characters such as punctuation can be added

later to the messages being stored. This six-bit scheme is convenient for storing in three colors, but also gives room to grow later if necessary.

Here's an example of the encoding of a message. Suppose "HELLO WORLD" was the message. Using the 28-integer code, it would be: 8, 5, 12, 12, 15, 27, 23, 15, 18, 12, 4, 0.

Create a method to encode the string into these integers. Type the following code into Steganography.java:

```
/**
 * Takes a string consisting of letters and spaces and
 * encodes the string into an arraylist of integers.
 * The integers are 1-26 for A-Z, 27 for space, and 0 for end of
 * string. The arraylist of integers is returned.
 * @param s string consisting of letters and spaces
 * @return ArrayList containing integer encoding of uppercase
 *      version of s
 */
public static ArrayList<Integer> encodeString(String s)
{
   s = s.toUpperCase();
   String alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   ArrayList<Integer> result = new ArrayList<Integer>();
   for (int i = 0; i < s.length(); i++)
   {
      if (s.substring(i,i+1).equals(" "))
      {
         result.add(27);
      }
      else
      {
         result.add(alpha.indexOf(s.substring(i,i+1))+1);
      }
   }
   result.add(0);
   return result;
}
```

Another method is necessary to decode a list of integers back into a string. Make the method `public` to test but then make it `private`.

```
/**
 * Returns the string represented by the codes arraylist.
 * 1-26 = A-Z, 27 = space
 * @param codes encoded string
 * @return decoded string
 */
public static String decodeString(ArrayList<Integer> codes)
{
   String result="";
   String alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   for (int i=0; i < codes.size(); i++)
   {
      if (codes.get(i) == 27)
      {
         result = result + " ";
      }
      else
      {
         result = result +
             alpha.substring(codes.get(i)-1,codes.get(i));
      }
   }
   return result;
}
```

To hide the string in the picture, one more method is needed. This method takes an integer between 0 and 63 and splits it into three two-bit pairs.

```
/**
 * Given a number from 0 to 63, creates and returns a 3-element
 *  int array consisting of the integers representing the
 *  pairs of bits in the number from right to left.
 * @param num number to be broken up
 * @return bit pairs in number
 */
private static int[] getBitPairs(int num)
{
   int[] bits = new int[3];
   int code = num;
   for (int i = 0; i < 3; i++)
   {
      bits[i] = code % 4;
      code = code / 4;
    }
   return bits;
}
```

2. You're now ready to create the method that hides the message. Complete the following method in the `Steganography` class.

```
/**
 * Hide a string (must be only capital letters and spaces) in a
 *  picture.
 *  The string always starts in the upper left corner.
 *  @param source picture to hide string in
 * @param s string to hide
 * @return picture with hidden string
 */
public static void hideText(Picture source, String s)
```

3. Now, complete the following method to be able to return the secret message hidden in the image.

```
/**
 * Returns a string hidden in the picture
 * @param source picture with hidden string
 * @return revealed string
 */
public static String revealText(Picture source)
```

4. In the `main` method, add code to test hiding and revealing a message.

Check Your Understanding

5. Given the representation scheme used, would it be possible to represent both uppercase and lowercase letters? What about digits? Exactly how many characters can be represented using the six-bit encoding scheme?

It would be possible to represent both uppercase and lowercase letters and digits. The six-bit scheme can have up to 64 different items!

6. When storing the secret message, a special value was used to signify the end of the message. Discuss with a partner what would happen if there was no way to signal the end of a message. Which methods would change? Describe how the behavior would be different.

The methods that would change is hideText() and revealText() and maybe any of the methods called from within! The behavior would be that the code never ends and be stuck in an infinite loop.