Objectives: Method Signatures and overriding methods
Returning early
Static variables.

MP3 re-graded tonight (/99%). MP4 due next Monday.

#1 Implement the Add method with three integer parameters
return the sum of the parameters unless
if any of the values are less than zero return zero
if the sum is greater than 1000 return 1000.

Implement an Add method with two integer parameters
with the same rules except that if the first value is -1 return -1.
Hint: Call your first Add method.

```
public class Program {
        public static void main(String[] args) {
                int a, b, c, sum;
                a = TextIO.getlnInt();
                b = TextIO.getlnInt();
                c = TextIO.getlnInt();
                sum = Add(a, b, c);
                System.out.println("Total is " + sum);
        }









}
```

Why can you not make a third method Add in the same class that takes two ints and returns a double? double Add(int x, int y)

#2 Create a ***class*** method *'debugArray'* that will print each value of a string array parameter.

#3 Where are the scoping errors in the following code? Can you fix them by adding parameters and return types?

```
public class Scope{
    private static final String BONJOUR = "hi"; // Class variable
    private static int count = 0; // Class variable

    public static void main(String [] args){
        int i = 6; // Local (temporary) variable only accessible inside main
        friendlyMethod();
        Scope.friendlyMethod();

        TextIO.putln(hello);
    }
    public static void friendlyMethod(){
        Scope.printer("Welcome");
        printer("Huh?");
        String hello = "hello!";
        printer(hello + Scope.BONJOUR);
        printer(hello + BONJOUR);
        TextIO.putln("i = " + i);
        count ++;
        TextIO.putln(count);
    }
     public static void printer(String h) {
       TextIO.putln(h + "…");
    }
}
```

Write a CLASS method that takes two parameters - a reference to a string array ("grades") and a string ("letter"). Return the array index of the grade letter, or -1 if the letter is not a valid grade letter.

What is `(0xff << 16) | (9 << 8) | 21` in hexadecimal?

```
class PixelEffects {
…
public static int[][]
       funky(int[][] source, int[][] sourceB) {

   int width = source.length;
   int height = source[0].length;
   int[][] result = new int[width][height];
   for (int i = 0; i < width; i++)
      for (int j = 0; j < height; j++) {
         int rgb = source[i][j];
         int red   = RGBUtilities.toRed(rgb);
         int green = RGBUtilities.toGreen(rgb);
         int blue  = RGBUtilities.toBlue(rgb);
         result[i][j] =
RGBUtilities.toRGB(0, Math.max(green,blue), 0);
      }
   return result;
}
```

Represent **Red-Green-Blue** Color Information as a single integer.

[http://math.hws.edu/javanotes/c13/s1.html#GUI2.1.2] The red, green, and blue components of a color are represented as 8-bit integers, in the range 0 to 255. When a color is encoded as a single int, the blue component is contained in the eight low-order bits of the int, the green component in the next lowest eight bits, and the red component in the next eight bits. (The eight high order bits store the "alpha component" of the color, which we'll encounter in the next section.) It is easy to translate between the two representations using the shift operators << and >> and the bitwise logical operators & and |.

Briefly: If A and B are integers, then A << B is the integer obtained by shifting each bit of A B bit positions to the left; A >> B is the integer obtained by shifting each bit of A B bit positions to the right; A & B is the integer obtained by applying the logical **and** operation to each pair of bits in A and B; and A | B is obtained similarly, using the logical **or** operation. For example, using 8-bit binary numbers, we have:

```
  01100101          01100101
& 10100001        | 10100001
  00100001          11100101
```

Here are incantations that you can use to work with color codes.

***Can you write an alternative version for green?**

```
/* Suppose that rgb is an int that encodes a color.
   To get separate red, green, and blue color components: *;

int red = (rgb >> 16) & 0xFF;
int green = (rgb >> 8) & 0xFF;
int blue = rgb & 0xFF;

/* Suppose that red, green, and blue are color components in
   the range 0 to 255.  To combine them into a single int: */

int rgb = (red << 16) | (green << 8) | blue;
```