CS 125 - Lecture 20

Objectives: Subroutines and 2-d array practice; Object-oriented next week!

Up next: MP4 due Monday, 8PM. Teach your music-major roommate about classes and subroutines. *http://digest.bps.org.uk/2014/10/students-learn-better-when-they-think.html*

1. Ask your neighbor … "**What is meant by pre-condition of a subroutine?"**

"…and post-condition?"

"What is a static variable?"

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

3. **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.**

4. **Complete the following bucket sort code to sort the data array.**

```
int[] data = {5,22,5,18,4,…. 74623 more values between 0 & 999

int max = 1000;
int[] histogram = new int[max];

// Phase 1, count the number of occurrences of 0,1,2,3... max-1

for(int i =0; i < data.length; i ++)

   ? _____________________________________________


// Phase 2, Use histogram to create the sorted output data

int ptr=0; // we will write values into data[ptr]

for(int value=0; value<max; value ++)

?

// This sort is fast but what limitations can you see with this
algorithm?
```

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

Can you think of an application where a `debugArrayInt()` **would be useful?**

6. **Funky town…**

```
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;
}
```

7. **Static Review -True/False?**

i) T / F? A common use of static variables is to create constants. For example,

```
public static final String MESG = "Abort? Retry? Fail?"
```

ii) T / F? Static methods are class methods.

iii) T / F? Static methods require an object.

iv) T /F? Class names usually start with an uppercase letter.

v) T / F? Method names usually start with a lowercase letter.

Correct the mistakes:

vi) T / F? You create static fields from inside a method.

e.g. To create a static variable you would write the following...

```
public class MyStringUtils {
    public static String addPadding(String s) {
     static char PADDINGCHAR =' ';
     while(s.length() < 10) s =  s + PADDINGCHAR;
     return s;
    }
}
```

vii) T / F? You call static methods using : '`ClassName.methodName(arguments)`'

8. **Using 2D arrays to represent an image.**

Create a picture of the JVMs memory and use memory pointers to explain why the following code swaps two rows.

```
int[][] pixels;
pixels = new int[480 /*row or 'y' coordinate*/][640 /* column or 'x'];
//  initialize pixel array : Odd rows are black.
//  Even rows are white
for(int y=0;y< 480; y++)
   for(int x = 0; x< 640; x++)
      if(y % 2 ==0) pixels[___][____] = 0xffffff;

//0xffffff = all white (red=255,green=255,blue=255)
int[] temp = pixels[10];
pixels[10] = pixels[11];
pixels[11] = temp;
```

CS 125 - Lecture 20

6. **Objects in your life? Examples:**

Tangible…

Abstract…

What characteristics do objects have in common?

6. **Merge…**

Complete the following code to merge two sorted integer arrays together into a single output array

```
public static int[] merge(int[] A, int[] B) {
   int done = 0;
   int countA = 0;
   int countB = 0;
   int[] result = new int[_____________________];
   while ((countA < A.length) ____ _____________________) {
      if (_____________________]) result[done++] = A[ countA++];
      else
         result[_______] = B[ __________];
   }
while (countA < A.length)
   result[done++] = A[countA++];
```

8. 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 or transparancy" of the color.) 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
 & 10100001
 ----------
   00100001
```

```
   01100101
 | 10100001
 ----------
   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;
```

where the binary representation of the integer 0xFF is:

```
0000 0000 0000 0000 0000 0000 1111 1111
```

most significant — least significant

```
/* 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;
```

7. **Thus far in this course, we have used classes to:**

8. **Classes can also be thought of as:**

9. **Objects as parameters in subroutines.** 2D Array Demo:

```
public class ArrayPassingDemo {

   public static void main(String[] args) {
      // test the array passing in here...
   }

   public static void debug2DArrayInt(int[][] ptr) {
      // Handle the case if the reference is null
      if (ptr == null) {
         System.out.println("Ooops! Array is null");
         return;
      }

      // Handle the case if the reference points to an array
      System.out.print("{ ");
      for (int i = 0; i < ptr.length; i++) {
         System.out.print("{ ");
         for (int j = 0; j < ptr[i].length; j++) {
            if (j > 0) System.out.print(", ");
            System.out.print(ptr[i][j]);
         }  // end of i-loop
         System.out.print(" } ");
      }  // end of j-loop
      System.out.println(" }");
   } // end of class method debug2DArrayInt
}
```