Section 11: Arrays

Introduction: An array is a common data structure: a way of efficiently organizing and storing related data. An array groups a set of pieces of data of the same variable type under a common name, with individual elements referenced by a numbered index. In lecture, we used buckets to represent the indices of the array and colored ping pong balls to represent the values stored in each index.

Arrays are useful in place of declaring many individual variables that will be used for similar purposes (e.g. the position variables from Lego Family). Arrays can be used and manipulated easily in conjunction with loops.

Array Declaration: Just like with variables, you must declare an array *before* you can use it. But unlike variables, array declarations come in two separate parts:

(1) Declaring the array variable is of the form: `var_type[] array_name;`

- `var_type` declares that your array will contain values of the specified data type.
- `array_name` is the name associated with the array. You should try to use an intuitive name that indicates what the array is used for. Has the same naming restrictions as other variables.

(2) Creating the array itself is of the form: `new var_type[len];`

- `var_type` sets the data type/container mold of every index and must match that of your array variable.
- `len` is the number of indices that Processing will create in your array (i.e. its length). Note that we always start counting indices from 0, so for `len` indices, they will be referenced as `0`, `1`, … , `len-1`.

Examples:

```
color[] myColors;           // declares array variable; no array created
int[] familyX = new int[5];  // creates array for x-positions of the
                             // 5 Simpsons characters
```

Array Initialization: Arrays differ from variables in that Processing will automatically initialize array values when you create a new array. The default value in an array depends on the data type, but roughly equates to "zero" in the various contexts: 0 (`int`), 0.0 (`float`), black (`color`), `false` (`boolean`).

Alternatively, you can initialize an array to chosen values using curly bracket (`{}`) notation, with values separated by commas. Every value must match the variable type of your array variable and the size of the array will depend on how many commas you use.

Examples:

```
color[] myColors = new color[3];         // all indices contain black
int[] familyX = {1, 100, 5, 200, 400};   // inferred array size of 5
```

Using Arrays: To use the values in an array, we use the array variable name along with brackets to indicate which index. The index can be specified using a hard-coded number or from the evaluation of an expression. As a reminder, array indices start at 0 and go to `len-1` (from left to right). The length of an array can be found using the special notation: `array_name.length`

Examples:

```
background( myColors[0] );          // set to the color in index 0
familyX[familyX[0]] = familyX[4];   // set familyX[1] to 400
int len = familyX.length;           // 5 based on initialization above
```

Exercises:

1) The following Processing statements contain errors. Find and fix them all.

Erroneous code:

```
int[3] intArray;
intArray = {0, 3.14, 6}
intArray[] = new int[7];
intArray[i+1;] = intArray[i];
```

Fixed code:

2) Write out a Processing statement below to declare and initialize an array that holds the colors of a tricolour flag of your choice (e.g. France, Germany, India, Mexico, Russia). Make sure that you give it an *intuitive* and *legal* array name.

3) Complete the function below that adds 1 to every index of an array of floats:

```
void addOne( float[] ar ) {
    int i = ___;
    while( i ___ __________ ) {
        _____________ ___ _____________;
        i ___ ___;
    }
}
```

4) Write out Processing code below to declare and initialize a length variable to 50, create an integer array of that length (using the variable), and then use a loop to initialize the array values to their indices (i.e. index 0 holds value 0, index 1 holds value 1, etc.).

5) Describe in a sentence what you think the following function accomplishes. Hint: make a simple test array and see what this function does to it!

```
void mystery( int[] ar ) {
    int temp, front = 0, back = ar.length - 1;
    while( front < back ) {
        temp = ar[front];
        ar[front] = ar[back];
        ar[back] = temp;
    }
}
```

6) Go to the course website and get started on the lab titled “Arrays and Elli.” [*partners*]