Reminder of selected arithmetic operations by example:

```java
int a = 14;
int b = 3;
int c = a / b;  // does integer division: answer is 4
int d = a % b;  // modulus operator: the remainder of the integer division problem a / b: answer is 2
```

Reminder of some array operations by example:

```java
int[] nums = new int[10];        // creates an array with of 10 ints.  
                                // all elements are initialized to 0
int[] squares = {0, 1, 4, 9, 16}; // an array of 5 integers with 
                                // initial values
System.out.println(nums.length); // number of elements (output: 10)
                                // valid indices are 0 – 9
String s = Arrays.toString(squares); // returns the string: 
                                // "[0, 1, 4, 9, 16]"
nums[1] = 54; // sets value at index 1 to 54
int save = nums[1]; // get value at index 1
int[] nums2 = nums; // two vars reference the same array
```

Note: array parameters are passed like objects: the array reference is passed by value, but the array itself is not copied. Thus, we can change values in an array from inside a method. For example:

```java
public static void update(int[] a) {
    a[2] = 20;
}
public static void main(String[] args) {
    int[] arr = new int[5];
    update(arr);
    System.out.println(a[2]);  // prints 20
}
```

**compareTo() String operation:**

```java
s1.compareTo(s2)
```

For Strings s1 and s2 in the call above, `compareTo` returns an int value either greater than zero, less than zero, or equal to zero, depending on which one has the greater lexicographic value (effectively, alphabetical comparison for words):

- `< 0` s1 comes before s2 in an alphabetical ordering
- `> 0` s1 comes after s2 in an alphabetical ordering
- `== 0` s1 equals s2 (i.e., `s1.equals(s2) == true`)