Upload your source code to blackboard when you have completed all exercises.

Arrays – preliminaries.

An array stores multiple values of same type, e.g.

```
String[] days = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
    "Saturday", "Sunday"};
```

Variables within an array are called *elements* and must be of the same data type (type `String` in the above example). You access the elements in an array through an index (i.e. an index is a number that identifies a specific element within an array), e.g.

```
System.out.println(days[0]); // "Monday"
```

Note that the indexing begins at 0. The length of an array can be obtained via its `length` constant, e.g.

```
int numDays = days.length; // 7
```

Note that there are no parentheses after the `length` constant. Do not confuse the array `length` public field constant with the `String` `length` method. Arrays may also be declared without explicitly initialising its elements, e.g.

```
int[] dailyValues = new int[100]; // declare an integer array of length 100
```

Arrays are objects. The variable denoting a given array stores a reference to the object, not the object itself. Therefore, unlike primitive data type variables which are *passed by value*, if an array is passed as an argument to a method, the method has ***direct access*** to the array and any changes made to the array elements within the method will change the elements themselves, e.g.

```
import java.util.Arrays;

public class ArrayMethodArguments {

   public static void main(String[] args) {
      int[] nums = {1, 2, 3, 4};
      setValue(nums[0]); // int value is passed to method
      System.out.println(Arrays.toString(nums)); // [1, 2, 3, 4]
      setElementValue(nums); // reference to array is passed to method
      System.out.println(Arrays.toString(nums)); // [0, 2, 3, 4]
   }

   public static void setValue(int value) {
      value = 0;
   }

   public static void setElementValue(int[] values) {
      values[0] = 0;
   }
}
```

For a similar reason, it is not possible to print the contents of an array by simply including the identifier in a `print` statement. Instead, use the `toString` method of the `Arrays` class. To use this method, the `Arrays` class must be imported into your program, e.g.

```
import java.util.Arrays;

public class ShowArray {
   public static void main(String[] args) {
      int[] nums = {1, 2, 3, 4};
      System.out.println(nums); // [I@7aca2076
      System.out.println(Arrays.toString(nums)); // [1, 2, 3, 4]
   }
}
```

Q.1

Write a static method named `swap` that accepts an array of integers and two indices and swaps the array elements at those indices. For example, if variable `a1` refers to an array containing {`12, 34, 56`}, the call `swap(a1, 1, 2)` should result in `a1` referring to the array {`12, 56, 34`}.

Test your code with the following class:

```
import java.util.Arrays;

public class Ex01TestSwap {
   public static void main(String[] args) {
      int[] a1 = {12, 34, 56};
      System.out.println("a1 contains " + Arrays.toString(a1));
      System.out.println("Swapping the element 1 and 2 ...");
      swap(a1, 1, 2);
      System.out.println("a1 contains " + Arrays.toString(a1));
   }
   // your method goes here
}
```

Q.2

Write a static method `swapAll` that accepts two arrays of integers as parameters and swaps their entire contents. For example, if variables `a1` and `a2` refer to arrays containing {`12, 34, 56`} and {`20, 50, 80`}, the call `swapAll(a1, a2)` should result in `a1` referring to the array {`20, 50, 80`} and `a2` referring to the array {`12, 56, 34`}.

Test your code with the following class:

```
import java.util.Arrays;

public class Ex02TestSwapAll {
   public static void main(String[] args) {
      int[] a1 = {12, 34, 56};
      System.out.println("a1 contains " + Arrays.toString(a1));
      int[] a2 = {20, 50, 80};
      System.out.println("a2 contains " + Arrays.toString(a2));
      System.out.println("Swapping a1 and a2 ...");
      swapAll(a1, a2);
      System.out.println("a1 contains " + Arrays.toString(a1));
      System.out.println("a2 contains " + Arrays.toString(a2));
   }
   // your method goes here
}
```

Q.3

Write a static method `merge` that accepts two arrays of integers and returns a new array containing all elements of the first array followed by all elements of the second. For example, if variables `a1` and `a2` refer to arrays containing {`12, 34, 56`} and {`20, 50, 80`}, the call `merge(a1, a2)` should result in an array containing {`12, 34, 56, 20, 50, 80`}.

Test your code with the following class:

```
import java.util.Arrays;

public class Ex03TestMerge {
   public static void main(String[] args) {
      int[] a1 = {12, 34, 56};
      System.out.println("a1 contains " + Arrays.toString(a1));
      int[] a2 = {20, 50, 80};
      System.out.println("a2 contains " + Arrays.toString(a2));
      int[] a3 = merge(a1, a2);
      System.out.println("Merging a1 and a2 into a3 ...");
      System.out.println("a3 contains " + Arrays.toString(a3));
   }
   // your method goes here
}
```

Q.4

Write a static method named `count` that accepts an array of integers and a target integer value as parameters and returns the number of occurrences of the target value in the array. For example, if a variable named `list` refers to an array containing values {`3, 5, 2, 1, 92, 38, 3, 14, 5, 73, 5`}, then the call of `count(list, 3)` should return `2` because there are two occurrences of the value `3` in the array.

Test your code with the following class:

```
public class Ex04TestCount {
   public static void main(String[] args) {
      int[] list = {3, 5, 2, 1, 92, 38, 3, 14, 5, 73, 5};
      System.out.println(count(list, 3)); // 2
      System.out.println(count(list, 5)); // 3
      System.out.println(count(list, 42)); // 0
   }
   // your method goes here
}
```

Q.5 - Optional

Write a static method named `isSorted` that takes an array of real numbers as a parameter and returns `true` if the list is in sorted (non-decreasing) order and `false` otherwise. For example, if variables named `list1` and `list2` refer to arrays containing {`16.1, 12.3, 22.2, 14.4`} and {`1.5, 4.3, 7.0, 19.5, 25.1, 46.2`}, respectively, the calls of `isSorted(list1)` and `isSorted(list2)` should return `false` and `true`, respectively. Assume the array has at least one element. A one-element array is considered to be sorted.

Test your code with the following class:

```
public class Ex05TestIsSorted {
   public static void main(String[] args) {
      double[] a1 = {16.1, 25.3, 12.2, 44.4};
      double[] a2 = {1.5, 4.3, 7.0, 19.5, 25.1, 46.2};
      double[] a3 = {42.0};
      System.out.println(isSorted(a1)); // false
      System.out.println(isSorted(a2)); // true
      System.out.println(isSorted(a3)); // true
   }
   // your method goes here
}
```