**Lab 7 Part1** (20 points)  
  
The purpose of this lab is to continue learning the BlueJ Integrated Development Environment (IDE) and provide hands on practice for Chapter 7 concepts. There are several learning objectives to this assignment

* Writing Class and Method Definitions with a focus on Arrays.
* Incorporating Information Hiding and Encapsulation
* Creating Constructors
* Using debugging techniques
* Incorporating documentation and style into your code

**Use the following UML to create a class called ArrayMethods. ArrayMethods will need to use the java.util.Arrays class (requires import).**

|  |
| --- |
| **ArrayMethods** |
| - a: int [ ] (initialized to {7, 8, 8, 3, 4, 9, 8, 7}) |
| + count (): int  + sum (): int  + average (): double  +findLast (int key): int  +findMax (): int  +findMin (): int  +findIndexOfMax (): int  +range (): int (use FindMin() and FindMax() to calculate range())  +findAll(int key): int []  +print(int [] a): void //provided below |
| +copyArray (): int [] *(Note: Implement Arrays.copyOf() to form a return type method)*  +sortArray (int [ ] inArray): int [ ]  +getArray(): int[]  +reverse(int[] inArray): int[]  +genFiveByFiveRows():void //generates int[5][5] and then prints int[][]  +genFiveByFiveCols():void //generates char[5][5] and then prints char[][] |

1. count () returns the number of values that are in the array  
   (Note: Use a for-each statement in this method to compute the number of values)
2. sum () returns the sum of the values that are in the array
3. average () returns the average of the array as a double
4. findLast () returns the index of the last value of the array based on the index parameter that is passed by value or -1 if the value ‘key’ is not found
5. findMax () returns the value of the largest integer in the array
6. findMin () returns the value of the smallest integer in the array
7. findIndexOfMax () returns the index of the value of the largest integer in the array
8. range () returns the difference between the largest and smallest number in int array.
9. findAll() creates and returns a new array containing the index(es) of every occurrence of a target value. Return an empty array of length 0 that contains nothing if the target value does not occur. Use the following array below as a print method to print the resultant findAll [ ]

*Hints: use 2 loops for this. The first counts how many times the target occurs. Next*

*create a new array, to hold this many indexes. The second loop puts the*

*indexes into the new array.*

Lab 7Pt2 (20pts)

Methods 10-15 will be part of Lab7Pt2.

1. getArray() is a simple getter for int[] a instance var.
2. sortArray() arranges a copy of int [ ] a into ascending order.   
     
   *Hints: You will want to use two for loops for this, one nested in the other and will need to use a temporary value to swap the value being evaluated with the value at an index if the value at the index is less than the value being evaluated.*
3. copyArray() is a creates a copy of the array passed using the Arrays.copyOf(). This is useful since Arrays.copyOf() is a static vs. return type method.
4. reverse() arranges the inputted array in reverse order. inArray[0] becomes inArray[inArray.length-1].
5. genFiveByFiveRows() generates and prints out a 5x5 int array. The columns are A->E and the rows are 1->5. For this method, row 1 will have all ones (1’s), row 2 will have all two’s (2’s), row 3 all three’s(3’s), row 4 all four’s (4’s), and row 5 all five’s (5’s).
6. genFiveByFiveCols() generates and prints out a 5x5 char array. The columns are A->E and the rows are 1->5. For this method, col 1 will have all A’s, col 2 will have all B’s, col 3 all C’s, col 4 all D’s, col 5 all E’s.

Other items of interest in ArrayMethods class

1. Use Arrays.copyOf() to create copyArray.
2. Arrays.sort() is the built in Java method for sorting an array
3. Arrays.equals () is the built in Java method for determining if two arrays are equal. Returns true if the two specified arrays of ints are *equal* to one another. Two arrays are considered equal if both arrays contain the same number of elements, and all corresponding pairs of elements in the two arrays are equal. In other words, two arrays are equal if they contain the same elements in the same order.

/\*

prints an int array, nicely formatted

@param a[] int array to print

\*/

public void print(int a[])

{

System.out.print("{");

int i;

// print elements before the last, separated by commas

for (i = 0; i < a.length - 1; ++i)

System.out.print(a[i] + ", ");

// print last element. Careful here to handle length 0

if (a.length > 0)

System.out.print(a[i]);

System.out.println("}");

}

You will need to create a driver for Part 1 that does the following:

1. Create an ArrayMethods object
2. A print statement that outputs the number of elements of the instance array in ArrayMethods class
3. A print statement that outputs the sum of each of the elements of the instance array in ArrayMethods class
4. A print statement that outputs the average of each of the elements of the instance array in ArrayMethods class
5. A print statement that outputs the last index of 8 (per findLast() description above.
6. A print statement that outputs the last index of 2 (per findLast() description above.
7. A print statement that outputs the max number in int[] a (per findMax() description above.
8. A print statement that outputs the index of the max number in int[] a (per findIndexOfMax() description above.
9. A print statement that outputs the range between the largest and smallest number of int[] a.
10. A statement that creates invokes findAll() based on the argument 8 and assigns this to a new array
11. A print statement that prints out the array created in the above step.

You will need to add the following steps to driver completed in Part1 to test the methods created for Part 2.

1. Preserve a copy of the original int[]a by using copyArray() – per above description – and assigns the copy of int[] a to a new int[]
2. A statement that prints whether Array a and the array copied in the previous step are equal using the Java Arrays.equal(). Pitfall: Make sure that you look up Arrays.equal() to see what arguments need to be passed in!
3. A statement that sorts the array created in Pt2 Step1 using the sortArray() per above description.
4. A print statement that outputs the array sorted in the previous step.
5. Repeat Part2 step 2 – equals test.
6. Repeat Part2 step1 and create another copy of int[] a and assign to another int[]. Note you should have three int[] in play. int[] a that is a ArrayMethods instance var, the copy of int[] a created in Pt2 Step1 and another copy of int[] a created in this step.
7. A print statement that outputs the array sorted in the previous step.
8. Sort the array created in Pt2 Step6 using the java Arrays.sort().
9. A print statement that prints out the int[] that was sorted in Pt2 Step8.
10. Repeat Part2 Step2 – equals test.
11. A statement that reverses the array created in Pt2 Step6
12. A print statement that prints out the int[] that was just reversed
13. A statement that invokes the genFiveByFiveRows()
14. A statement that invokes the genFiveByFiveCols()