Lab #8
CSE110 - Arizona State University

Topics

- Basic arrays

Coding Guidelines

- Give identifiers semantic meaning and make them easy to read (examples numStudents, grossPay, etc).
- Keep identifiers to a reasonably short length.
- Use upper case for constants. Use title case (first letter is upper case) for classes. Use lower case with uppercase word separators for all other identifiers (variables, methods, objects).
- Use tabs or spaces to indent code within blocks (code surrounded by braces). This includes classes, methods, and code associated with ifs, switches and loops. Be consistent with the number of spaces or tabs that you use to indent.
- Use white space to make your program more readable.
- Use comments after the ending brace of classes, methods, and blocks to identify to which block it belongs.

Assignment/Lab Documentation

At the beginning of each programming assignment you must have a comment block with the following information:

```
/*-------------------------------------------------------------------------
// AUTHOR: your name
// FILENAME: title of the source file
// SPECIFICATION: description of the program
// FOR: CSE 110- Lab #8
// TIME SPENT: how long it took you to complete the assignment
//-----------------------------------------------------------------------*/
```

Getting Started

Create a class called `Lab8`. Use the same setup for setting up your class and main method as you did for the previous assignments. Be sure to name your file `Lab8.java`.

Additionally, make another file called `Arrays.java`. This file will be an object, so simply start it off by declaring an `Arrays` class. You can copy the following skeleton and fill in the appropriate code below each of the comments:

```
public class Arrays {
     // Instance Variables
     // Constructors
     // findMin
```

```
    // findMax
    // calcSum
    // calcAverage
    // toString
}
```

Hints

- (Page 250) An array will be used in this program.
- (Page 258) Section 6.3 contains some array algorithms used in this lab.
- The lecture and code example videos are also good references for this lab.

Task Overview

Your task for this lab is to create a class called `Arrays` with some array processing methods. This class will maintain an array and the number of elements present in it. Additionally, methods will be available to display the current min and max elements along with the average of all of them. Finally, a `toString()` method will be available to cleanly display all the array elements.

Finally, you will write a simple driver class to test out the above `Arrays` class.

Part 1: Instance Variables for Arrays

The first thing to do for the `Arrays` class is to set up its instance variables. Declare the following (`private`) instance variables:

- An `int` array called `array` ? this will be the array we will be writing methods for.
- An `int` called `count` - this represents the number of valid elements in the array.

Part 2: Constructors for Arrays

The `Arrays` class will have two constructors. The first constructor takes the maximum size of the `array` as input as a parameter and initializes the `array` instance variable appropriately. It also sets `count` to `size`. Finally, it will initialize all of the array elements to some values between 0 and 10, inclusive. To create this constructor, follow these steps:

- Import `java.util.Random` to make use of the random number generator.
- Create a constructor with the following header: `public Arrays(int size)`
- Initialize your `array` variable and set its size to `size` (see the chart on page 252 for reference on initializing arrays). Be very careful that you are setting the value of your `array` instance variable, as opposed to creating a new variable called `array`.
- Set the value of the `count` variable to `size` because we will be populating the entire array.
- Copy the following code to the constructor in order to generate random values between 0 and 10, inclusive:

```
Random rand = new Random();
for (int i = 0; i < count; i++) {
     array[i] = (rand.nextInt(10));
}
```

Next, create another constructor with the following header: `public Arrays(int[] arr)`. This constructor will initialize the class by using the passed `arr` argument in order to fill its instance variables. The following things need to be done inside of this constructor:

- Set the `array` variable equal to `arr`.
- Set the `count` variable equal to the `length` of the `array`.

Part 3: Displaying the Output

findMin()

The first method of this class will search the array for the minimum element. Copy the following code for the `findMin` method. Note how the `count` instance variable is used instead of `array.length`. This is just in case the entire array is not being used (it will be in our case, though).

```
public int findMin() {
     int min = array[0]; // Set min to the first element
     for (int i = 1; i < count; i++) {
          // Reassign min if there is a smaller element
          if (array[i] < min) {
               min = array[i];
          }
     }
     return min; // Return the smallest element
}
```

findMax()

Using the above code as reference, write a method which finds the maximum element within the array. You can refer to page 259 if you are stuck.

calcSum()

The `calcSum()` method will be a `private` method (for clarity, it is the only `private` method in this class). It will be used later on within the class as a helper method, but never outside of the class. This method will return the sum of all of the elements in the array. You can use the following steps as guidelines for how to complete this method, if you choose:

- Use the following as the header: `private int calcSum()`
- Declare an `int` variable for the `sum` - initialize it to 0
- Write a `for` loop to iterate through all the elements in the array (remember to use the `count` instance variable instead of the `length` of the `array`).
- Add the value of each element to the `sum`.
- Return the `sum` after the `for` loop

You can refer to page 259 if you are stuck.

calcAverage()

This method will return the average of all of the elements in the array. Use the following for the header of this method: `public double calcAverage()`. **IMPORTANT**: Use the `calcSum()` `private` method in your computation of the average for full credit. Hints:

- You will need to make use of the total elements in the `array`.
- You will need to cast the result to a `double` at the appropriate time to achieve an accurate average.

toString()

The `toString()` method is called whenever an object is passed into a print statement. This particular `toString()` method will print the following, assuming the array consists of elements `{1, 2, 3, 4}`:

```
[ 1, 2, 3, 4 ]
```

Copy the following code to use for this method:

```
public String toString() {
     String output = "[ ";
     for (int i = 0; i < count; i++) {
          output += array[i];
          if (i != count - 1) {
               output += ", ";
          }
     }
     return output + " ]";
}
```

Part 4: Test Class for Arrays

At this point, the `Arrays` class is completed. The next step is to create a driver class to test it. This is the `Lab8.java` file that you created at the beginning of the lab. Copy the following code into the `main` method of the `Lab8.java` file. This code tests the first constructor of the `Arrays` class along with all its methods.

```
// Create an Arrays object using the first constructor
Arrays arr1 = new Arrays(5);
// Print the contents of the array in arr1
System.out.println(arr1);
// Call findMin, findMax, and calcAverage on arr1 and print their values
System.out.println("Min: " + arr1.findMin());
System.out.println("Max: " + arr1.findMax());
System.out.println("Average: " + arr1.calcAverage());
System.out.println();
```

The next step is to add code which tests the second constructor of the `Arrays` class. To do this, complete the following tasks by adding code to the end of the code you just copied into the `main` method.

- Create an `int` array of length 3 - explicitly set its values to any 3 ints by using an array initialization list. See the chart on the bottom of page 252 for a reference on how to set an array with initial values.
- Create an `Arrays` object using the second constructor. Note that this involves passing the `array` variable you just created. Call this object `arr2`.
- Print the `arr2` object by passing it into a `println` statement.
- Print the min, max, and average of the `arr2` object, as what was done with the `arr1` object.

Sample Output

Below is an example of what your output should roughly look like when this lab is completed. Please note that your values will almost certainly be different, depending both on the random number generator and the values of the array you created in `Lab8.java`. The following run initialized its array in the `arr2` object to contain `{1, 2, 3}`.

Sample Run:
`[2, 1, 8, 4, 4]`
Min: 1
Max: 8
Average: 3.8

`[1, 2, 3]`
Min: 1
Max: 3
Average: 2.0

Submission

Submit your `Lab8.java` to the Submission Server. Go to the Submission Server site located on the course website, login, then click on Lab Submissions in the left frame.Choose Lab8 from the dropdown box, click on the browse button and find where you saved your `Lab8.java` file (and not the `Lab8.class` file) on your computer. Do the same for `Arrays` (again, NOT `Arrays.class`). Upload the files to the site and then click on the Submit button.

Your file will be submitted and a screen will show up displaying if your program compiled and what your output is when run on some sample input.

You should then check to make sure that the actual file submitted properly and is readable to the grader. To do so click on Grades in the frame on the left of the page and then click on the 0 underneath Lab8. You will again see that your program compiled and the sample output, but you should scroll down to the bottom of the screen and make sure your file is readable as well.

Important Note: You may resubmit as many times as you like until the deadline, but we will only mark your last submission.