Intro to Java Week 3 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

1. Create an array of int called ages that contains the following values: 3, 9, 23, 64, 2, 8, 28, 93.
    a. Programmatically subtract the value of the first element in the array from the value in the last element of the array (i.e. do not use ages[7] in your code). Print the result to the console.
    b. Add a new age to your array and repeat the step above to ensure it is dynamic (works for arrays of different lengths).
    c. Use a loop to iterate through the array and calculate the average age. Print the result to the console.
2. Create an array of String called names that contains the following values: "Sam", "Tommy", "Tim", "Sally", "Buck", "Bob".
    a. Use a loop to iterate through the array and calculate the average number of letters per name. Print the result to the console.
    b. Use a loop to iterate through the array again and concatenate all the names together, separated by spaces, and print the result to the console.

3. How do you access the last element of any array?

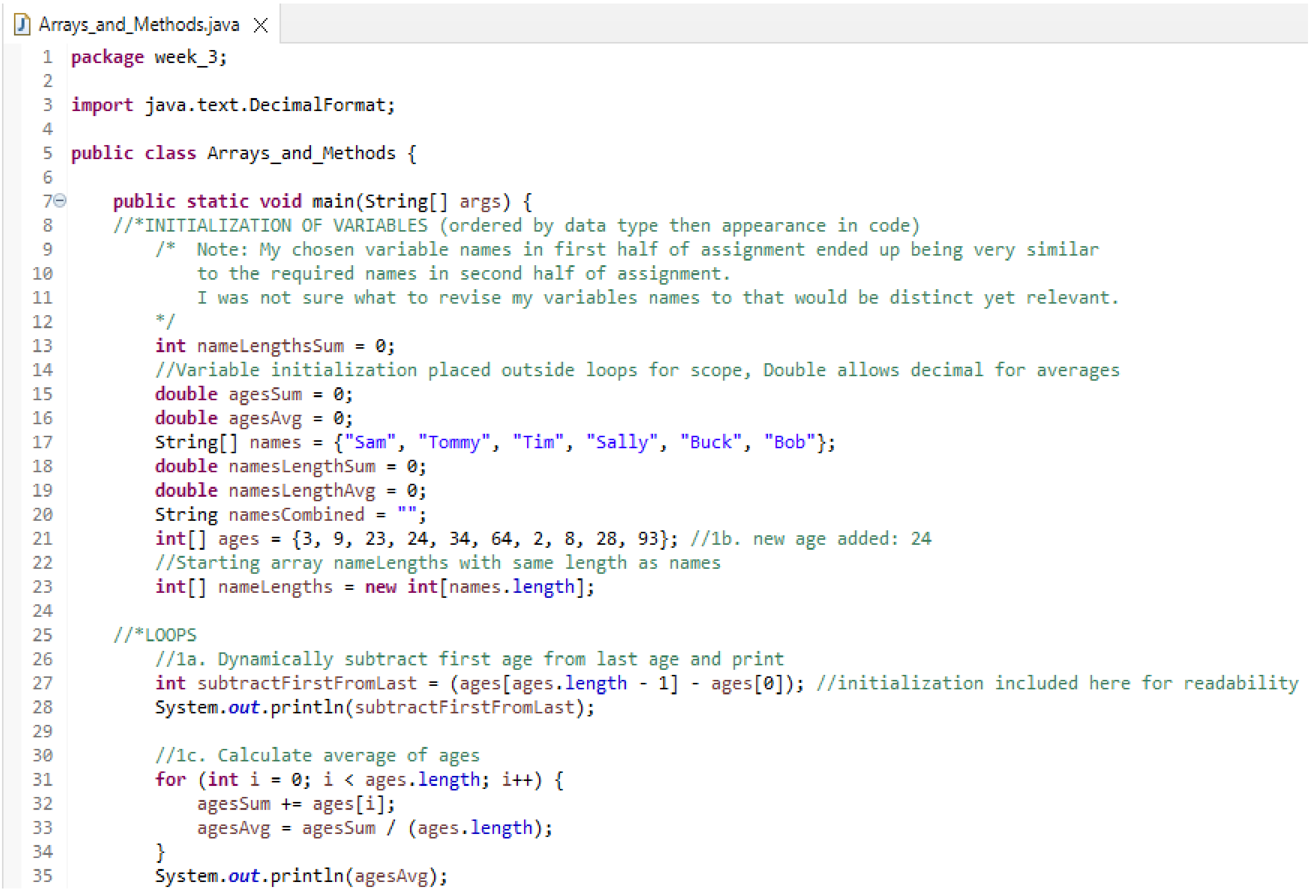
    arr[array.length - 1];

4. How do you access the first element of any array?

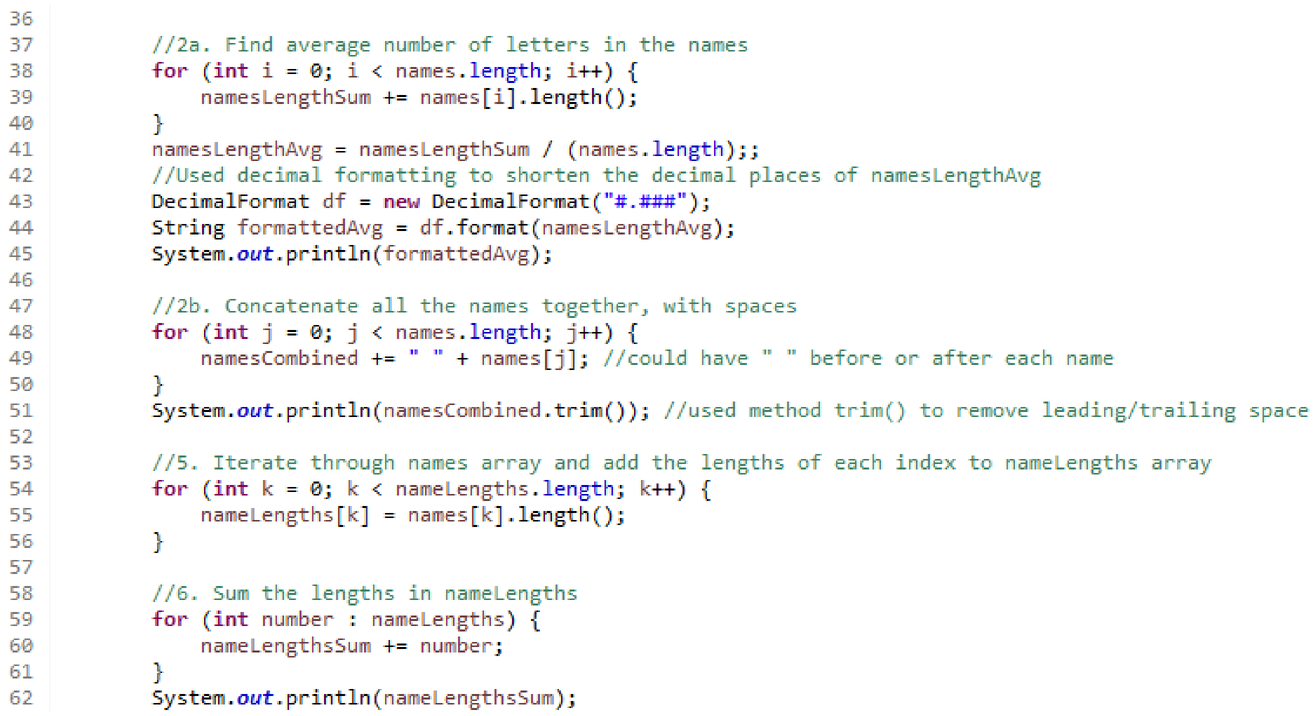
    arr[0];

5. Create a new array of int called nameLengths. Write a loop to iterate over the previously created names array and add the length of each name to the nameLengths array.
6. Write a loop to iterate over the nameLengths array and calculate the sum of all the elements in the array. Print the result to the console.
7. Write a method that takes a String, word, and an int, n, as arguments and returns the word concatenated to itself n number of times. (i.e. if I pass in “Hello” and 3, I would expect the method to return “HelloHelloHello”).
8. Write a method that takes two Strings, firstName and lastName, and returns a full name (the full name should be the first and the last name as a String separated by a space).
9. Write a method that takes an array of int and returns true if the sum of all the ints in the array is greater than 100.
10. Write a method that takes an array of double and returns the average of all the elements in the array.
11. Write a method that takes two arrays of double and returns true if the average of the elements in the first array is greater than the average of the elements in the second array.
12. Write a method called willBuyDrink that takes a boolean isHotOutside, and a double moneyInPocket, and returns true if it is hot outside and if moneyInPocket is greater than 10.50.
13. Create a method of your own that solves a problem. In comments, write what the method does and why you created it.

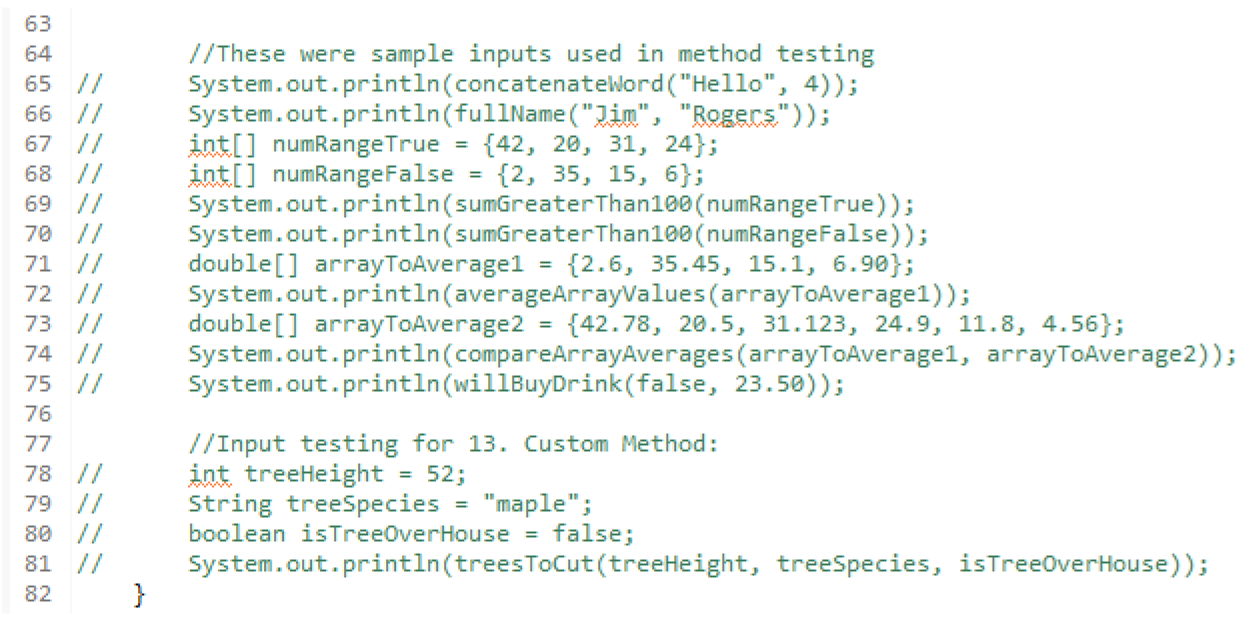
Screenshots of Code:



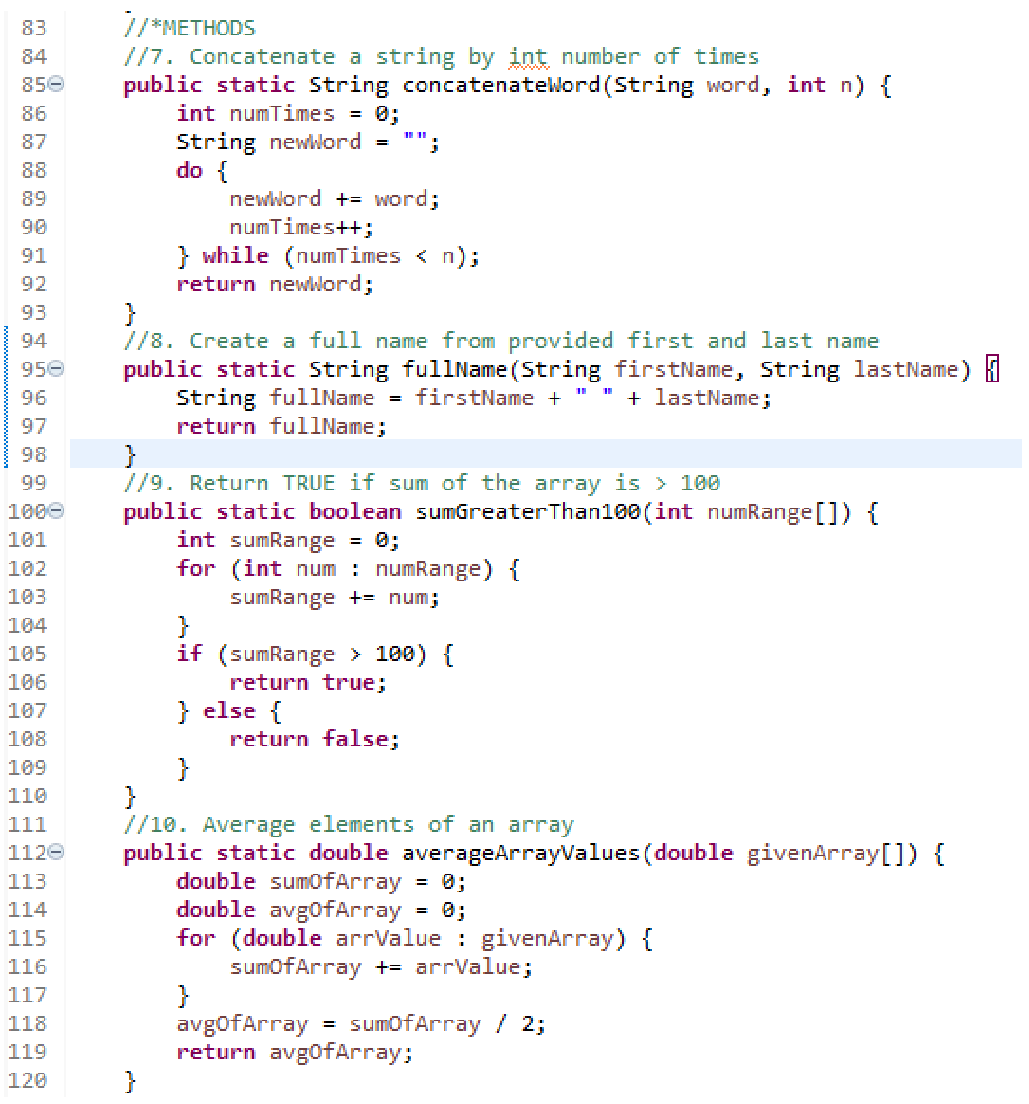
```java
package week_3;

import java.text.DecimalFormat;

public class Arrays_and_Methods {

    public static void main(String[] args) {
    //*INITIALIZATION OF VARIABLES (ordered by data type then appearance in code)
        /*  Note: My chosen variable names in first half of assignment ended up being very similar
            to the required names in second half of assignment.
            I was not sure what to revise my variables names to that would be distinct yet relevant.
        */
        int nameLengthsSum = 0;
        //Variable initialization placed outside loops for scope, Double allows decimal for averages
        double agesSum = 0;
        double agesAvg = 0;
        String[] names = {"Sam", "Tommy", "Tim", "Sally", "Buck", "Bob"};
        double namesLengthSum = 0;
        double namesLengthAvg = 0;
        String namesCombined = "";
        int[] ages = {3, 9, 23, 24, 34, 64, 2, 8, 28, 93}; //1b. new age added: 24
        //Starting array nameLengths with same length as names
        int[] nameLengths = new int[names.length];

    //*LOOPS
        //1a. Dynamically subtract first age from last age and print
        int subtractFirstFromLast = (ages[ages.length - 1] - ages[0]); //initialization included here for readability
        System.out.println(subtractFirstFromLast);

        //1c. Calculate average of ages
        for (int i = 0; i < ages.length; i++) {
            agesSum += ages[i];
            agesAvg = agesSum / (ages.length);
        }
        System.out.println(agesAvg);
```



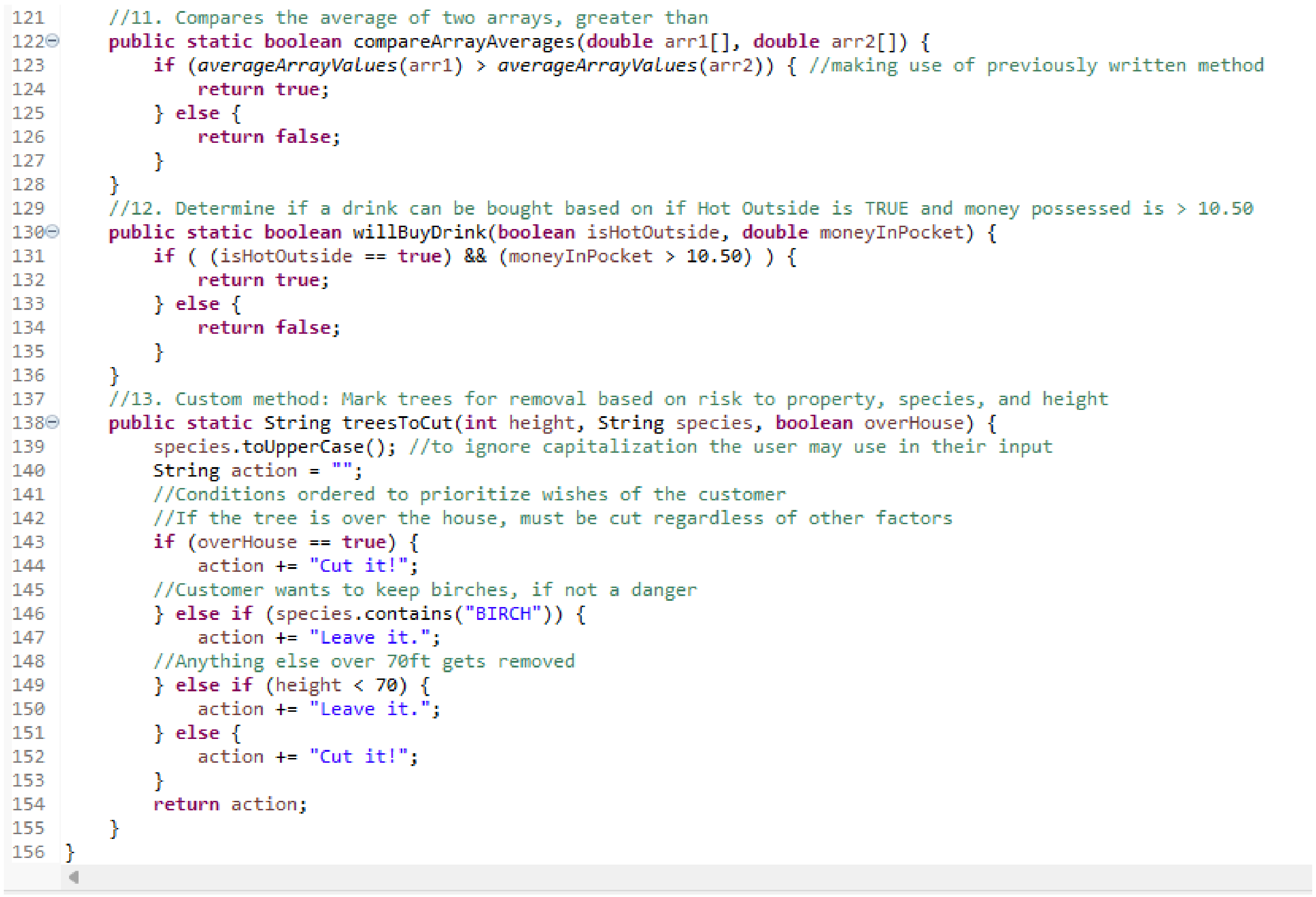
```java

        //2a. Find average number of letters in the names
        for (int i = 0; i < names.length; i++) {
            namesLengthSum += names[i].length();
        }
        namesLengthAvg = namesLengthSum / (names.length);;
        //Used decimal formatting to shorten the decimal places of namesLengthAvg
        DecimalFormat df = new DecimalFormat("#.###");
        String formattedAvg = df.format(namesLengthAvg);
        System.out.println(formattedAvg);

        //2b. Concatenate all the names together, with spaces
        for (int j = 0; j < names.length; j++) {
            namesCombined += " " + names[j]; //could have " " before or after each name
        }
        System.out.println(namesCombined.trim()); //used method trim() to remove leading/trailing space

        //5. Iterate through names array and add the lengths of each index to nameLengths array
        for (int k = 0; k < nameLengths.length; k++) {
            nameLengths[k] = names[k].length();
        }

        //6. Sum the lengths in nameLengths
        for (int number : nameLengths) {
            nameLengthsSum += number;
        }
        System.out.println(nameLengthsSum);
```



```java

        //These were sample inputs used in method testing
//      System.out.println(concatenateWord("Hello", 4));
//      System.out.println(fullName("Jim", "Rogers"));
//      int[] numRangeTrue = {42, 20, 31, 24};
//      int[] numRangeFalse = {2, 35, 15, 6};
//      System.out.println(sumGreaterThan100(numRangeTrue));
//      System.out.println(sumGreaterThan100(numRangeFalse));
//      double[] arrayToAverage1 = {2.6, 35.45, 15.1, 6.90};
//      System.out.println(averageArrayValues(arrayToAverage1));
//      double[] arrayToAverage2 = {42.78, 20.5, 31.123, 24.9, 11.8, 4.56};
//      System.out.println(compareArrayAverages(arrayToAverage1, arrayToAverage2));
//      System.out.println(willBuyDrink(false, 23.50));

        //Input testing for 13. Custom Method:
//      int treeHeight = 52;
//      String treeSpecies = "maple";
//      boolean isTreeOverHouse = false;
//      System.out.println(treesToCut(treeHeight, treeSpecies, isTreeOverHouse));
    }
```

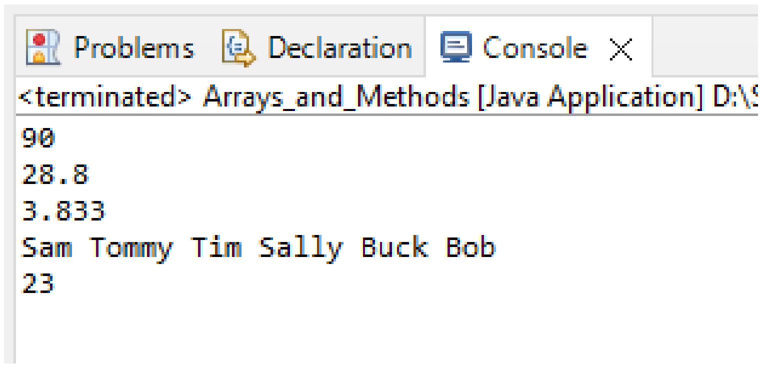


```java
    //*METHODS
    //7. Concatenate a string by int number of times
    public static String concatenateWord(String word, int n) {
        int numTimes = 0;
        String newWord = "";
        do {
            newWord += word;
            numTimes++;
        } while (numTimes < n);
        return newWord;
    }
    //8. Create a full name from provided first and last name
    public static String fullName(String firstName, String lastName) {
        String fullName = firstName + " " + lastName;
        return fullName;
    }
    //9. Return TRUE if sum of the array is > 100
    public static boolean sumGreaterThan100(int numRange[]) {
        int sumRange = 0;
        for (int num : numRange) {
            sumRange += num;
        }
        if (sumRange > 100) {
            return true;
        } else {
            return false;
        }
    }
    //10. Average elements of an array
    public static double averageArrayValues(double givenArray[]) {
        double sumOfArray = 0;
        double avgOfArray = 0;
        for (double arrValue : givenArray) {
            sumOfArray += arrValue;
        }
        avgOfArray = sumOfArray / 2;
        return avgOfArray;
    }
```



```java
    //11. Compares the average of two arrays, greater than
    public static boolean compareArrayAverages(double arr1[], double arr2[]) {
        if (averageArrayValues(arr1) > averageArrayValues(arr2)) { //making use of previously written method
            return true;
        } else {
            return false;
        }
    }
    //12. Determine if a drink can be bought based on if Hot Outside is TRUE and money possessed is > 10.50
    public static boolean willBuyDrink(boolean isHotOutside, double moneyInPocket) {
        if ( (isHotOutside == true) && (moneyInPocket > 10.50) ) {
            return true;
        } else {
            return false;
        }
    }
    //13. Custom method: Mark trees for removal based on risk to property, species, and height
    public static String treesToCut(int height, String species, boolean overHouse) {
        species.toUpperCase(); //to ignore capitalization the user may use in their input
        String action = "";
        //Conditions ordered to prioritize wishes of the customer
        //If the tree is over the house, must be cut regardless of other factors
        if (overHouse == true) {
            action += "Cut it!";
        //Customer wants to keep birches, if not a danger
        } else if (species.contains("BIRCH")) {
            action += "Leave it.";
        //Anything else over 70ft gets removed
        } else if (height < 70) {
            action += "Leave it.";
        } else {
            action += "Cut it!";
        }
        return action;
    }
}
```

Screenshots of Running Application:



```
<terminated> Arrays_and_Methods [Java Application] D:\
90
28.8
3.833
Sam Tommy Tim Sally Buck Bob
23
```

URL to GitHub Repository:

https://github.com/tylerjlivermore/Week3_ArraysAndMethods.git