

2D Array & Array Copy and Enhanced Loop

1. Introduction

This report presents two distinct programming activities. The first activity focuses on the development of a Java program designed to calculate and display various academic averages for a class. The second activity explores array manipulation techniques, specifically the enhanced for loop and array copying methodologies. These activities utilize fundamental programming concepts, including data structures, control flow, and method utilization.

1.1. Activity 1 – 2D Array

The primary objective of this activity is to create a Java program that processes mid-term grades for a class. The program will manage grades for two subjects: Math and Science. The input data consists of a set of paired grades representing the scores of individual students in both subjects. Specifically, the program will calculate the average score for each subject (Math and Science) and the overall class average based on the provided grades.

To achieve this, the program will employ a two-dimensional array to store the grade data. This array will be initialized with the provided grade values, eliminating the need for user input during program execution. The structure of the array is designed to organize the math and science grades efficiently for calculation. The program will then leverage the data stored within the array to calculate the desired averages.

The implementation of the average calculations will be structured using two static methods. The first method will calculate and return the class average. The second method will calculate and return the average for each subject, Mathematics and Science. Within these

methods, nested for loops will be used to iterate through the two-dimensional array. These loops will facilitate the summation of grades for each subject and the overall class, which are then used in the calculation of the averages. The final output will be formatted to display the calculated averages with a precision of two decimal places, ensuring the results are presented in a clear and concise manner.

1.1. Lab 1 - Array Copy and Enhanced Loop

The second part of this report provides an exploration into array manipulation in Java, with a focus on array copying and the use of enhanced for loops. The enhanced for loop, also known as the “for-each” loop, offers a simplified approach to iterating over arrays. The program demonstrates the use of the enhanced for loop with a simple integer array, iterating through each element and printing its value.

The concept of copying arrays is then introduced, contrasting a reference copy with a true copy. The report explains how assigning one array reference variable to another creates a reference copy, where both variables point to the same array in memory. Any changes made through one reference will be reflected in the other. To make a true copy, where a separate instance of the array is created and the elements are copied individually, a standard for loop is employed to copy the contents of an existing array into a new array. This method ensures the independence of the two arrays. The program then shows an example demonstrating the difference between a reference copy and a true array copy and how to copy one array into another.

2. Background

This report addresses the use of arrays in Java programming to store and manipulate data. Specifically, it explores the concepts of two-dimensional arrays and the enhanced for loop, alongside methods for copying arrays. A two-dimensional array, as used in Activity 1, provides a structure to organize data in rows and columns, such as the math and science grades provided. This structure allows for efficient storage and retrieval of related data, in this case, grades for multiple subjects. The report will demonstrate how a two-dimensional

array can be initialized and used to perform calculations, specifically calculating the average grades for math and science, as well as the overall class average. The program is designed to meet specific requirements, including the use of nested loops for calculations and the formatting of output to two decimal places.

The report further delves into the enhanced for loop, a simplified method for iterating through arrays. The syntax is `for (dataType elementVariable : array)`. It iterates through each element in an array, providing a more concise way to access each array value. The limitations of using an enhanced for loop are also addressed.

Finally, the report explores the concept of array copying, demonstrating the difference between a reference copy and a true copy. A reference copy does not create a new array but assigns the same memory address to multiple array variables, so changes made through one variable will affect the other. A true copy involves creating a new array and copying the elements from the original array using a loop, ensuring that modifications to one array do not impact the other.

3. Source Code Explanation

3.1. Activity 1 – 2D Array

3.1.1. Average_Calc.java

```
public class Average_Calc {  
  
    // calculating the class average  
  
    public static double calculateClassAverage(double[][] grades) {  
  
        double total = 0; // initialize total variable
```

The `Average_Calc` class is declared, and within it, the `calculateClassAverage` method is defined. This method is designed to compute the average of all grades provided. Inside the

method, a variable named `total` is initialized to zero. This variable will accumulate the sum of all the grades.

```
for (int i = 0; i < grades.length; i++) {  
  
    for (int j = 0; j < grades[i].length; j++) {  
  
        total += grades[i][j];  
  
    }  
  
}  
  
return total / (grades.length * 2);
```

Nested loops are used to iterate through a two-dimensional array named `grades`. The outer loop iterates through the rows of the array, and the inner loop iterates through the columns of each row. In the inner loop, each grade `grades[i][j]` is added to the total. After processing all grades, the method calculates the class average by dividing the total by the total number of grades which is obtained by multiplying the number of rows (`grades.length`) by the number of columns (which is presumed to be 2). The method then returns this calculated average.

```
// calculating the average of math and science subject  
  
public static double[] calculateSubjectAverages(double[][] grades) {  
  
    double mathTotal = 0; // initialize math total variable  
  
    double scienceTotal = 0; // initialize science total variable
```

The `calculateSubjectAverages` method is defined to compute the average grade for math and science separately. Two variables, `mathTotal` and `scienceTotal`, are initialized to zero. These variables will store the sum of the math grades and science grades, respectively.

```
for (int i = 0; i < grades.length; i++) { // loop to iterate through each subject grades
```

```
mathTotal += grades[i][0];

scienceTotal += grades[i][1];

}

return new double[] { mathTotal / grades.length, scienceTotal / grades.length };
```

A loop iterates through the rows of the grades array. Inside the loop, the math grade (at index 0 of each row) is added to *mathTotal*, and the science grade (at index 1 of each row) is added to *scienceTotal*. After the loop, the method calculates the average math grade by dividing *mathTotal* by the number of rows (*grades.length*) and the average science grade by dividing *scienceTotal* by the number of rows. Finally, a new array containing the math and science averages is created and returned.

```
public static void main(String[] args) {

    double[][] grades = { // 2D array of grades for math and science subjects

        {80.5, 25.5},

        {100, 92.5},

        {75.0, 65.5},

        {82.3, 79.6},

        {35.9, 22.7},

        {73.6, 66.2},

        {88.9, 98.7},

        {46.2, 62.3},

        {100, 100},

        {97.8, 99.5}
```

```
};

    double classAvg = calculateClassAverage(grades); //calling the method to calculate
class average

    double[] subjectAverages = calculateSubjectAverages(grades); // calling the method
to calculate subject averages
```

The main method is the entry point of the program. A two-dimensional array named grades is declared and initialized within the main method. This array contains the grades for each student, organized with the math score in the first column and the science score in the second column. The array holds data for ten students. Following the data definition, the calculateClassAverage method is called, passing the grades array as an argument. The result, the class average, is stored in the variable classAvg. Subsequently, the calculateSubjectAverages method is called, also with the grades array, and the returned array containing the subject averages is stored in subjectAverages.

```
// displaying the results

    System.out.printf("Class Average: %.2f\n", classAvg);

    System.out.printf("Math Average: %.2f\n", subjectAverages[0]);

    System.out.printf("Science Average: %.2f\n", subjectAverages[1]);
```

The code then displays the calculated results. The printf method is used to output formatted strings to the console. The class average (classAvg) is displayed, formatted to two decimal places. Then, the math average (at index 0 of subjectAverages) and the science average (at index 1 of subjectAverages) are displayed, both formatted to two decimal places. The program's execution then completes.

3.2. Lab 1 - Array Copy and Enhanced Loop

3.2.1. CopyArray.java

```
public class CopyArray {
```

```
public static void main(String[] args) {  
  
    int ARRAY_SIZE = 5;  
  
    int[] array1 = {2, 4, 6, 8, 10};  
  
    int[] array2 = new int[ARRAY_SIZE];
```

This code snippet initializes a class named copyArray. Inside the main method, it declares an integer constant ARRAY_SIZE and sets its value to 5. Then, an integer array named array1 is initialized with the values 2, 4, 6, 8, and 10. Another integer array, array2, is created with a size determined by ARRAY_SIZE.

```
// make array2 references a copy of array1  
  
    for (int index = 0; index < array1.length; index++) {  
  
        array2[index] = array1[index];  
  
    }
```

This section iterates through the elements of array1 and copies each element's value to the corresponding index in array2. The for loop uses an index variable to access each element. The comment explains the purpose of this code block: to make array2 a copy of array1.

```
// change one of the element using array1  
  
    array1[0] = 200;  
  
    // change one of the elements using array2  
  
    array2[4] = 1000;
```

Here, the value of the element at index 0 in array1 is changed to 200. Subsequently, the value of the element at index 4 in array2 is changed to 1000.

```
// display all of the elements using array1
```

```
System.out.println("The contents of array1:");

for (int value : array1) {

    System.out.print(value + " ");

}

System.out.println();
```

This part of the code displays the contents of array1 to the console. It first prints a descriptive message “The contents of array1:”. Then, an enhanced for loop iterates through each value in array1, printing each value followed by a space. Finally, a newline character is printed.

```
// display all of the elements using array2

System.out.println("The contents of array2:");

for (int value : array2) {

    System.out.print(value + " ");

}

System.out.println();

}

}
```

This section displays the contents of array2 to the console, following the same pattern as the display for array1. It prints a descriptive message, iterates through the array using an enhanced for loop, prints the values, and adds a newline character at the end. The “}” closes the main method and the “}” closes the class.

3.2.2. EnhancedFor.java

```
public class EnhancedFor {
```



```
public static void main(String[] args) {  
  
    int[] numbers = {3, 6, 9};  
  
    for (int val : numbers)  
  
    {  
  
        System.out.println(val);  
  
    }  
  
}  
  
}
```

The enhancedFor class is defined. Inside the main method, an integer array named numbers is initialized with the values 3, 6, and 9. An enhanced for loop is used to iterate through the elements of the numbers array. In each iteration, the current element's value is assigned to the variable val. The value of val is then printed to the console.

3.2.3. ReferenceCopy.java

```
public class ReferenceCopy {  
  
    public static void main(String[] args) {  
  
        int[] array1 = {2, 4, 6, 8, 10};  
  
        int[] array2 = array1;
```

The referenceCopy class is declared. The main method initializes an integer array array1 with the values 2, 4, 6, 8, and 10. Then, array2 is assigned the value of array1. This means array1 and array2 now reference the same array in memory.

```
// change one of the element using array1  
  
array1[0] = 200;
```

```
// change one of the element using array2
```

```
array2[4] = 1000;
```

Here, the value of the element at index 0 in array1 is changed to 200. Subsequently, the value of the element at index 4 in array2 is changed to 1000. Because array1 and array2 refer to the same array, changes made through one variable will be reflected in the other.

```
// display all os the ements of array1
```

```
System.out.println("The contents of array1:");
```

```
for (int value : array1) {
```

```
    System.out.print(value + " ");
```

```
}
```

```
System.out.println();
```

This segment displays the contents of array1 to the console. It begins by printing “The contents of array1: ”. An enhanced for loop is then used to iterate through the array, printing each element followed by a space. Finally, a newline character is printed.

```
// display all os the ements of array2
```

```
System.out.println("The contents of array2:");
```

```
for (int value : array2) {
```

```
    System.out.print(value + " ");
```

```
}
```

```
System.out.println();
```

```
}
```

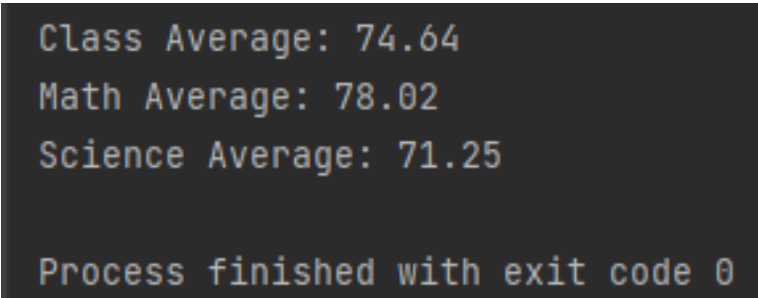
```
}
```

This code block displays the contents of array2 to the console, following the same pattern as the display for array1. It prints a descriptive message, iterates through the array using an enhanced for loop, prints the values, and adds a newline character at the end. The “}” closes the main method and the “}” closes the class.

4. Results and Discussion

4.1. Activity 1 – 2D Array

The program calculated the class averages for Math and Science, as well as the overall average for the entire class. The results showed that the Math average was 78.02, while the Science average was slightly lower at 71.25. The overall class average came out to 74.64, which falls between the two subject averages. This suggests that, on average, students performed better in Math than in Science. The small difference between the two subject averages indicates that while the class was somewhat stronger in Math, the performance in Science was not far behind. The use of a two-dimensional array made it easy to organize and process the grades, and the nested loops efficiently handled the calculations. The formatted output, displaying the results to two decimal places, made the data clear and easy to interpret.



```
Class Average: 74.64
Math Average: 78.02
Science Average: 71.25

Process finished with exit code 0
```

4.2. Lab 1 - Array Copy and Enhanced Loop

The outputs from the array manipulation activities clearly demonstrated the difference between a copy and a reference copy of an array. In modifying array1 (changing the first element to 200) did not affect array2, and vice versa (changing the last element

of array2 to **1000**). This confirmed that the two arrays were independent, as expected when creating a true copy.

```
The contents of array1:
200 4 6 8 10
The contents of array2:
2 4 6 8 1000

Process finished with exit code 0
```

However, in the reference copy, both array1 and array2 pointed to the same memory location. When the first element of array1 was changed to 200, it also reflected in array2. Similarly, modifying the last element of array2 to **1000** updated array1 as well. This behavior highlighted the key difference between reference copies (where changes affect both arrays) and true copies (where changes remain isolated). The enhanced for loop also proved useful for cleanly iterating through arrays without manual index management.

```
The contents of array1:
200 4 6 8 1000
The contents of array2:
200 4 6 8 1000

Process finished with exit code 0
```

The enhanced for loop demonstrated a simple and efficient way to iterate through arrays. When the program ran, it printed each value in the numbers array (3, 6, and 9) on separate lines. This output confirmed that the enhanced for loop correctly accessed and displayed each element without needing manual index handling.

```
3
6
9

Process finished with exit code 0
```

5. Conclusion

This report details the implementation and analysis of two Java programs related to array manipulation and data processing. The first program calculates and presents the average scores for math and science, as well as the overall class average, based on provided midterm grades. The second focuses on array copying techniques and the use of the enhanced for loop.

The core functionality of the first program centers on a two-dimensional array. This structure efficiently stores the math and science grades. The program uses a nested loop to iterate through the array, calculating the average math score, average science score, and the overall class average. The results are then displayed, formatted to two decimal places, as shown in the sample run. The sample run provided demonstrated the program's ability to calculate the average scores, showing the class average, math average, and science average.

The second part of the report explores array copying, highlighting the difference between reference copies and the creation of independent arrays. The report introduces the enhanced for loop, a simplified method for iterating through arrays. Several examples are provided to demonstrate its use and its limitations. The program utilizes the enhanced for loop to print each value in the array. Then, a scenario that highlights a reference copy is analyzed where changes made to one array also impact the other. Finally, it illustrates the proper method for copying arrays: creating independent arrays with their own memory locations. This is achieved by looping through the elements of the original array and assigning their values to the corresponding elements of a new array. The output

demonstrates that modifying one array after the copy does not affect the other, thus confirming their independence.

The analysis of both programs indicates a successful implementation of the specified requirements. The average score calculation program demonstrates proficient use of two-dimensional arrays and nested loops, essential for organizing and processing the provided data. The array copying program successfully illustrates the difference between reference copies and creating independent arrays, offering valuable insights into array manipulation in Java. The use of the enhanced for loop provides a simpler way to iterate through arrays.