ASSIGNMENT

SUBMITTED TO: Mr Sajid Maqbool

SUBMITTED BY: Abdur-Rehman Munir

ROLL NO: 2K23-BSCS-524

SECTION: M

DEPARTMENT: COMPUTER SCIENCE

Session: 2K23 – 2K27

Semester: 3rd

Subject: Data Structures

NFC Institute of Engineering and Technology

Java Program to Initialize and Print a 2D Array of Prime Numbers

Introduction:

This Java program demonstrates how to initialize a 2D array with specific values (prime numbers in this case) and print the contents of the array in a structured format. The approach involves defining a function to handle the printing of the 2D array without relying on array length properties, making it adaptable to specified dimensions.

Code:

```
public class ArrayPrinter {

   public static void main(String[] args) {

         System.out.println("Name= Abdurrehman Munir"+" "+ "Roll no = 2K23-BSCS-524"+" "+"Section: M");


      // Initialize a 2D array with prime numbers
      int[][] array = {
         {2, 3, 5},
         {7, 11, 13},
         {17, 19, 23}
      };

      // Call the function to print the array
      print2DArray(array);
   }

   // Function to print a 2D array with fixed row and column values in the loop
   public static void print2DArray(int[][] array) {
      // Using fixed values (3 for rows and 3 for columns)
      for (int i = 0; i < 3; i++) {       // Row count
         for (int j = 0; j < 3; j++) {   // Column count
            System.out.print(array[i][j] + " ");
         }
         System.out.println();
      }
   }
}
```

Code Explanation

1. Class Declaration (ArrayPrinter)

- We define a class named ArrayPrinter to contain the main method and a helper function for printing the 2D array.

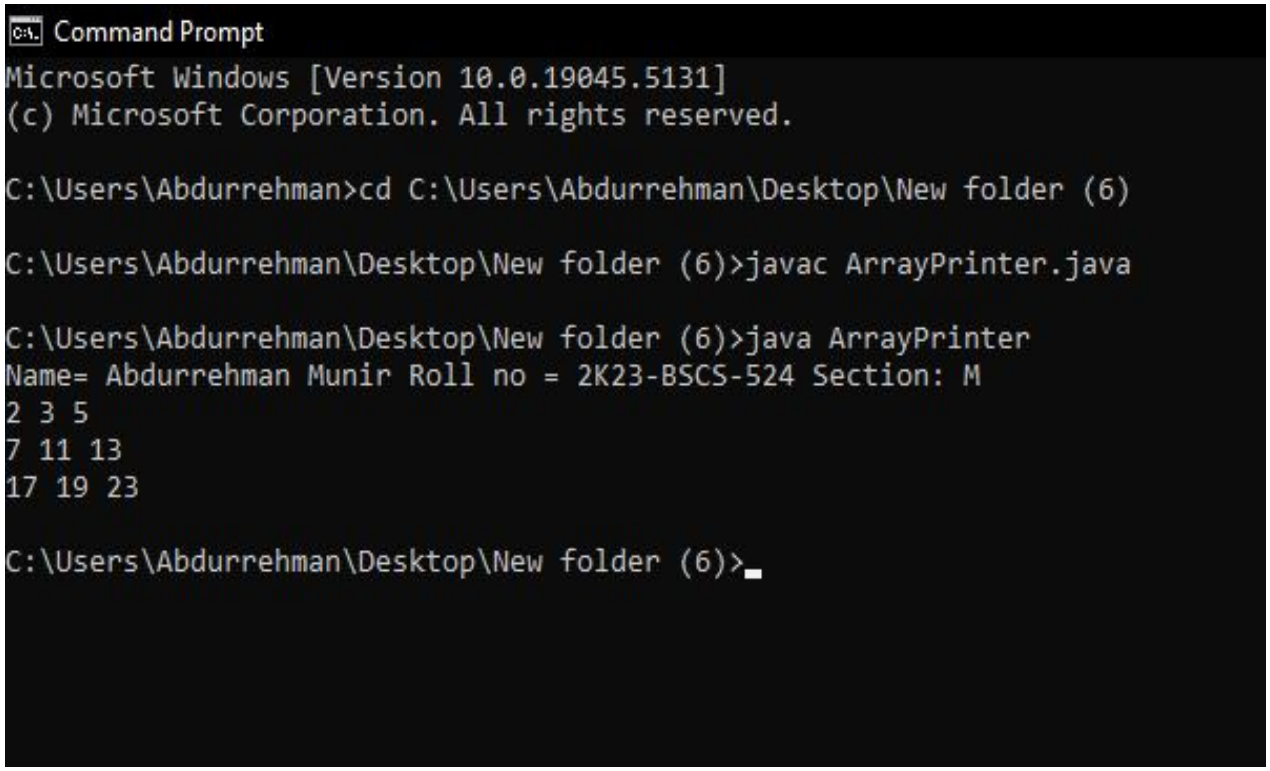
2. Main Method (public static void main(String[] args))

- **Array Initialization**: A 3x3 2D array, array, is initialized with the first nine prime numbers (2, 3, 5, 7, 11, 13, 17, 19, 23). Each element is unique and follows a logical pattern (prime sequence).
- **Function Call**: The print2DArray function is called with array as an argument. This function handles the task of iterating over each element and printing it in a structured way.

3. Print Function (public static void print2DArray(int[][] array))

- **Looping Through Rows and Columns**: The function uses nested for loops to access each row and column element. Here, we use fixed values 3 for rows and columns since we know the array is 3x3.
- **Printing Elements**: System.out.print(array[i][j] + " "); prints each element in the current row separated by a space.
- **New Line for Each Row**: After each row, System.out.println(); moves the output to a new line, creating a grid-like display of the array.

OUTPUT:



Output Explanation

When executed, this program produces a structured 3x3 grid of prime numbers:

Name= Abdurrehman Munir Roll no = 2K23-BSCS-524 Section: M
2 3 5
7 11 13
17 19 23

Each row and column are aligned neatly, representing the contents of the 2D array. This output helps visualize the data in a table-like format, which is particularly useful for understanding matrix structures or representing ordered sets of data in programming.

Example 2.2 Duplicating an Array

Java Program for Duplicating an Array Using clone() Method

Introduction

This Java program demonstrates how to duplicate an array by using the clone() method. Cloning an array creates a separate copy of the array's elements, which can be modified independently. This is particularly useful when working with both int and String arrays, as shown in this example.

Code:

```
public class DuplicatingArrays {
   public static void main(String[] args) {
      System.out.println("Name= Abdurrehman Munir"+" "+ "Roll no = 2K23-BSCS-524"+" "+"Section: M");

      int[] a = {22, 44, 66, 88};
      print(a);

      int[] b = (int[])a.clone();  // duplicate a[] in b[]
      print(b);

      String[] c = {"AB", "CD", "EF"};
      print(c);

      String[] d = (String[])c.clone();  // duplicate c[] in d[]
      print(d);

      c[1] = "XYZ";  // change c[], but not d[]
      print(c);
      print(d);
   }

   public static void print(int[] a) {
      System.out.printf("{%d", a[0]);
      for (int i = 1; i < a.length; i++) {
         System.out.printf(", %d", a[i]);
      }
```

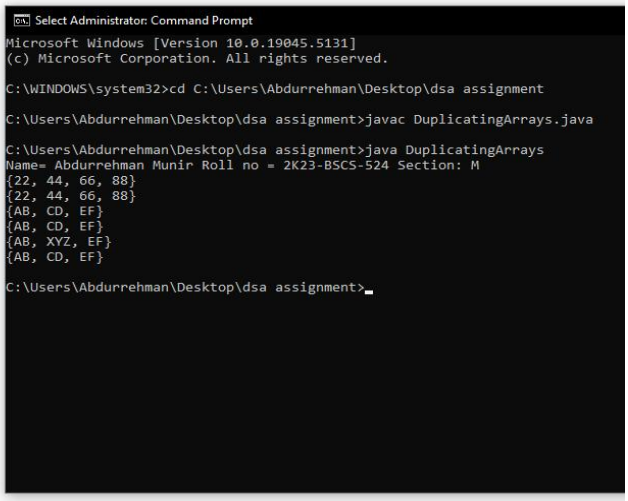
```
        System.out.println("}");
    }

    public static void print(Object[] a) {
        System.out.printf("{%s", a[0]);
        for (int i = 1; i < a.length; i++) {
            System.out.printf(", %s", a[i]);
        }
        System.out.println("}");
    }
}
```

Code with output:

DuplicatingArrays.java - Notepad

File Edit Format View Help

```
public class DuplicatingArrays {
    public static void main(String[] args) {
       System.out.println("Name= Abdurrehman Munir"+" "+ "Roll no = 2K23-BSCS-524"+" "+"Section: M");
        int[] a = {22, 44, 66, 88};
        print(a);

        int[] b = (int[])a.clone();  // duplicate a[] in b[]
        print(b);

        String[] c = {"AB", "CD", "EF"};
        print(c);

        String[] d = (String[])c.clone();  // duplicate c[] in d[]
        print(d);

        c[1] = "XYZ";  // change c[], but not d[]
        print(c);
        print(d);
    }

    public static void print(int[] a) {
        System.out.printf("{%d", a[0]);
        for (int i = 1; i < a.length; i++) {
            System.out.printf(", %d", a[i]);
        }
        System.out.println("}");
    }

    public static void print(Object[] a) {
        System.out.printf("{%s", a[0]);
        for (int i = 1; i < a.length; i++) {
            System.out.printf(", %s", a[i]);
        }
        System.out.println("}");
    }
}
```



Code Explanation

1. Class Declaration (DuplicatingArrays)

- A class named DuplicatingArrays contains the main method and helper functions for printing both int and String arrays.

2. Main Method (public static void main(String[] args))

- **Array Initialization and Printing**:
    - An integer array a is initialized with values {22, 44, 66, 88} and printed using print(a).
- **Cloning Integer Array**:
    - b is a cloned copy of a created using a.clone(). The clone() method creates a duplicate of a, allowing changes to one without affecting the other. print(b) is called to display the cloned array.
- **String Array Initialization and Cloning**:
    - A String array c is initialized with {"AB", "CD", "EF"} and printed using print(c).
    - A cloned copy d of c is created using c.clone(). print(d) is called to display d.
- **Modifying Original Array**:
    - The second element of c (c[1]) is changed to "XYZ". This demonstrates that d remains unaffected, showing the independence of the cloned array. print(c) and print(d) are called again to show the modified c and unchanged d.

3. Print Method for Integer Arrays (public static void print(int[] a))

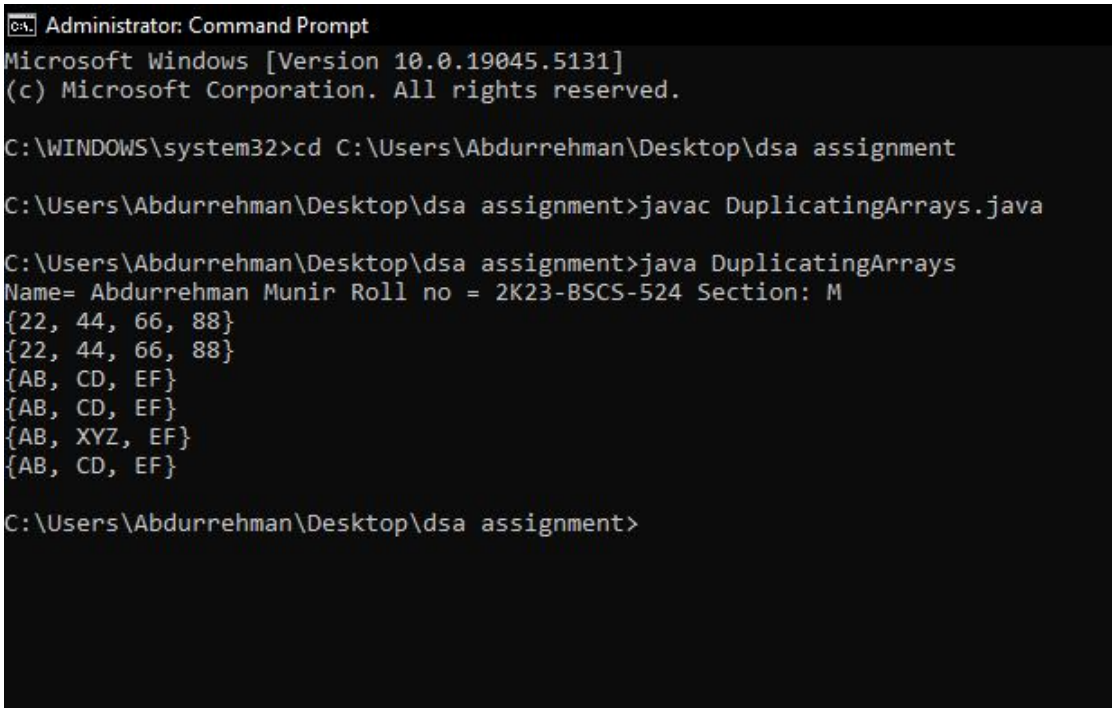
- This method takes an integer array a and prints its elements in {} brackets.
- A for loop iterates through each element, formatting the output to be comma-separated.

4. Print Method for Object Arrays (public static void print(Object[] a))

- This method takes an array of Object type (used here for String arrays) and prints the elements in {} brackets, similar to the integer array print function.
- A for loop formats the elements with commas, making the output visually consistent.

***Purpose**: It prints the contents of an object array (a) in the format {AB, CD, EF}, which is useful for arrays of String or other object types.*

OUTPUT:



Output Explanation:

Name= Abdurrehman Munir Roll no = 2K23-BSCS-524 Section: M
{22, 44, 66, 88}
{22, 44, 66, 88}
{AB, CD, EF}
{AB, CD, EF}
{AB, XYZ, EF}
{AB, CD, EF}

Detailed Output Analysis

1. {22, 44, 66, 88} – The original integer array a is printed.
2. {22, 44, 66, 88} – The cloned integer array b is printed, showing it's identical to a.
3. {AB, CD, EF} – The original String array c is printed.
4. {AB, CD, EF} – The cloned String array d is printed, identical to c.
5. {AB, XYZ, EF} – After modifying c[1] to "XYZ", c is printed again, showing the change.
6. {AB, CD, EF} – Finally, d is printed, showing it remains unaffected by the modification to c, confirming d is an independent copy of c.

Example:2.3 using java.util.Arrays class

Topic: Demonstrating Java java.util.Arrays Class and Array Operations

CODE:

```
import java.util.Arrays;

public class TestArrays {
   public static void main(String[] args) {
       System.out.println("Name=  Abdurrehman  Munir"+"
"+ "Roll no = 2K23-BSCS-524"+" "+"Section: M");
      int[] a = {44, 77, 55, 22, 99, 88, 33, 66};
      print(a);

      Arrays.sort(a);
      print(a);

      int k = Arrays.binarySearch(a, 44);
      System.out.printf("Arrays.binarySearch(a,
44): %d%n", k);
      System.out.printf("a[%d]: %d%n", k, a[k]);

      k = Arrays.binarySearch(a, 45);
      System.out.printf("Arrays.binarySearch(a,
45): %d%n", k);

      int[] b = new int[8];
      print(b);

      Arrays.fill(b, 55);
      print(b);

      System.out.printf("Arrays.equals(a,       b):       "       +
Arrays.equals(a, b));
   }
```

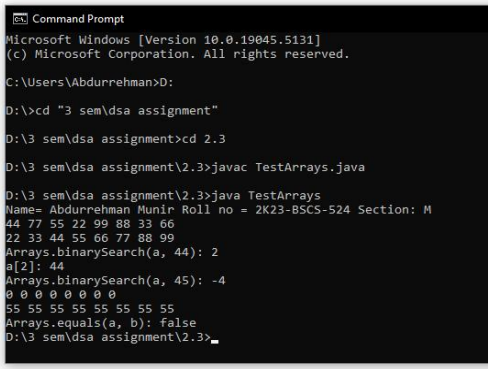
```
    public static void print(int[] a) {
        System.out.printf("%d", a[0]);
        for (int i = 1; i < a.length; i++) {
            System.out.printf(" %d", a[i]);
        }
        System.out.println();
    }
}
```

Code with output:

TestArrays.java - Notepad

File Edit Format View Help

```
import java.util.Arrays;

public class TestArrays {
    public static void main(String[] args) {
         System.out.println("Name= Abdurrehman Munir"+" "+ "Roll no = 2K23-BSCS-524"+" "+"Section: M");
        int[] a = {44, 77, 55, 22, 99, 88, 33, 66};
        print(a);

        Arrays.sort(a);
        print(a);

        int k = Arrays.binarySearch(a, 44);
        System.out.printf("Arrays.binarySearch(a, 44): %d%n", k);
        System.out.printf("a[%d]: %d%n", k, a[k]);

        k = Arrays.binarySearch(a, 45);
        System.out.printf("Arrays.binarySearch(a, 45): %d%n", k);

        int[] b = new int[8];
        print(b);

        Arrays.fill(b, 55);
        print(b);

        System.out.printf("Arrays.equals(a, b): " + Arrays.equals(a, b));
    }

    public static void print(int[] a) {
        System.out.printf("%d", a[0]);
        for (int i = 1; i < a.length; i++) {
            System.out.printf(" %d", a[i]);
        }
        System.out.println();
    }
}
```



Code Explanation

Displaying Program Information

The program starts by printing the user's name, roll number, and section as a header for the output. This is a simple use of concatenation in System.out.println.

2. Declaring and Printing the Original Array

An array a is initialized with values {44, 77, 55, 22, 99, 88, 33, 66}.

- The method print(a) is used to display the array elements in a single line, separated by spaces. The custom print method iterates over the array and prints each element.

3. Sorting the Array

The Arrays.sort(a) method is called to sort the array in ascending order.

- After sorting, the array a is printed again using the print(a) method, showing the rearranged elements.

4. Binary Search on the Sorted Array

The Arrays.binarySearch(a, value) method is used to perform a binary search on the sorted array.

- When searching for 44, it returns the index where 44 is found (e.g., index 2). The program displays both the index and the value at that index.
- When searching for 45, which does not exist in the array, the method returns a negative value (-insertion point - 1). This indicates where 45 would be inserted to maintain the sorted order.

5. Initializing a New Array

A new array b of size 8 is declared and initialized. By default, all elements are 0. The print(b) method is called to display its contents.

6. Filling the New Array with a Value

The Arrays.fill(b, 55) method is used to replace all elements of the array b with the value 55.

- The updated array is then printed, showing all elements as 55.

7. Comparing Two Arrays

The Arrays.equals(a, b) method checks if the two arrays a and b are identical in terms of size and element values.

- Since a and b have different elements, the result is false, which is printed in the output.

8. Custom Print Method

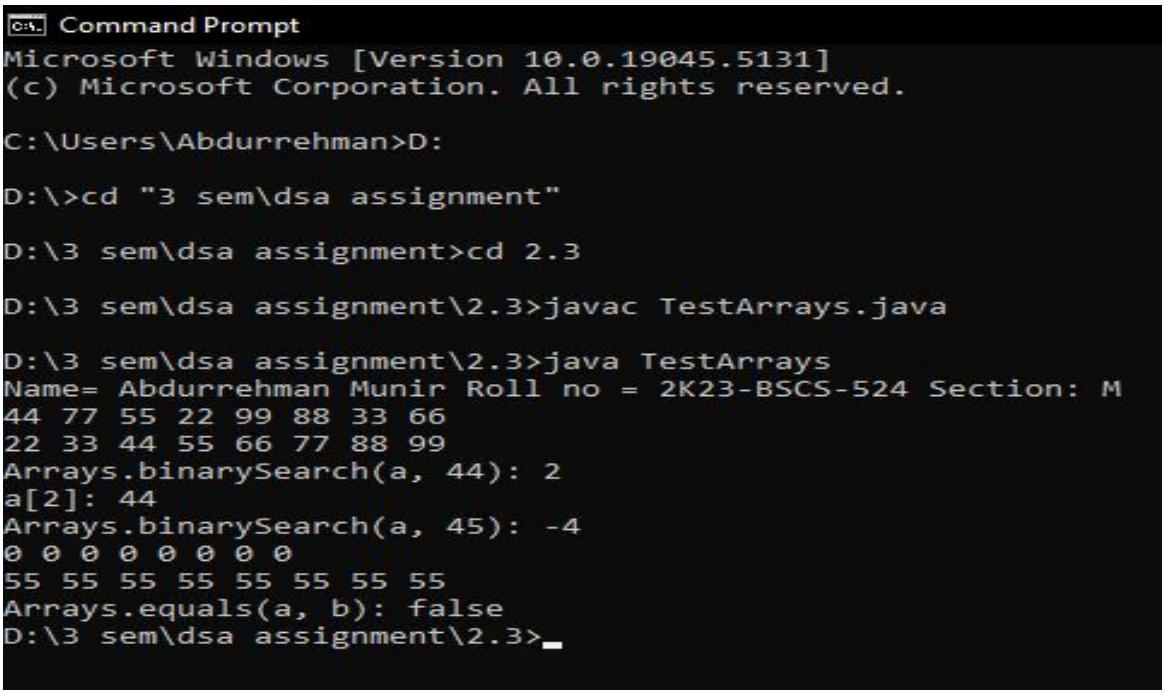
The custom print(int[] a) method is used to display the contents of any array passed to it. It iterates through the array using a loop and prints elements in a space-separated format. This avoids the need for repeatedly writing code to display arrays.

Summary of Key Operations

1. **Sorting:** Arrays.sort rearranges the array in ascending order.
2. **Binary Search:** Arrays.binarySearch locates the position of an element or determines where it should be inserted.
3. **Filling:** Arrays.fill populates all elements of an array with a specific value.
4. **Equality Check:** Arrays.equals compares arrays for identical size and content.
5. **Custom Display:** The print method provides a formatted view of the array's elements.

This program demonstrates how java.util.Arrays can be used for common array manipulations in Java efficiently.

OUTPUT



Code Output

Name= Abdurrehman Munir Roll no = 2K23-BSCS-524 Section: M
44 77 55 22 99 88 33 66
22 33 44 55 66 77 88 99
Arrays.binarySearch(a, 44): 2
a[2]: 44
Arrays.binarySearch(a, 45): -4
0 0 0 0 0 0 0 0
55 55 55 55 55 55 55 55
Arrays.equals(a, b): false

Explanation of Output:

Original Array

- **Code:** `print(a)`
- **Explanation:**
  This line displays the unsorted array a. It uses Arrays.toString() to convert the array into a readable string format.
  Example output: Original Array: [44, 77, 55, 22, 99, 88, 33, 66].

Sorted Array

- **Code:** Arrays.sort(a);

- **Explanation:**
  The Arrays.sort() method is used to sort the array a in ascending order. After sorting, a is displayed using Arrays.toString().
  Example output: Sorted Array: [22, 33, 44, 55, 66, 77, 88, 99].

Binary Search

- **Code:** int index = Arrays.binarySearch(a, 44);
- **Explanation:**
  The Arrays.binarySearch() method performs a binary search on the sorted array a to find the index of a specified value (in this case, 44).
    - **If the value exists:** Returns the index of the value in the sorted array. Example: Index of 44 is 2.
    - **If the value does not exist:** Returns -(insertion point) - 1. For 45, it returns -4, meaning it would be inserted at index 4.

New Array Initialization

- **Code:** int[] b = new int[8];
- **Explanation:**
  Creates a new array b of size 8. By default, an uninitialized int array contains all zeroes.
  Example output: New Array b (Initially): [0, 0, 0, 0, 0, 0, 0, 0].

Filling an Array

- **Code:** Arrays.fill(b, 55);
- **Explanation:**
  The Arrays.fill() method fills all elements of array b with the specified value (55).
  Example output: New Array b (After filling with 55): [55, 55, 55, 55, 55, 55, 55, 55].

Array Comparison

- **Code:** boolean areEqual = Arrays.equals(a, b);
- **Explanation:**
  The Arrays.equals() method compares the contents of two arrays. It returns true if both arrays are equal (same length and identical elements in the same order), otherwise false.
  Example output: Array Comparison: false (a and b have different values).