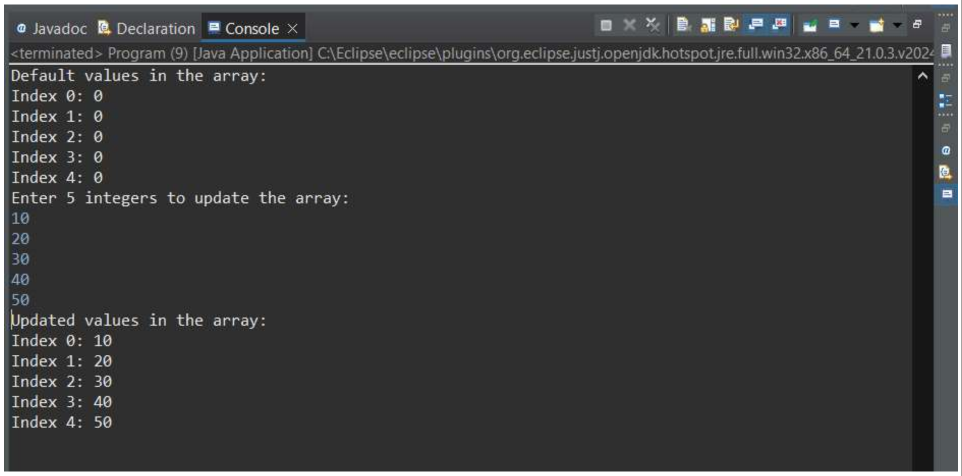
Ass 7 (SN)

Sunday, September 15, 2024 4:03 PM

1. Declare a single-dimensional array of 5 integers inside the `main` method. Traverse the array to print the default values. Then accept records from the user and print the updated values of the array.

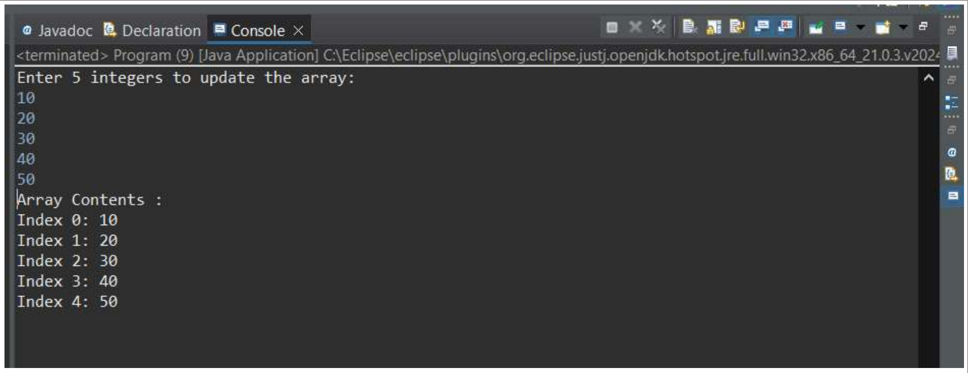
```java
package org.example.q1;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        // Declare a single-dimensional array of 5 integers
        int[] arr = new int[5];

        // Traverse and print the default values
        System.out.println("Default values in the array:");
        for (int i = 0; i < arr.length; i++) {
            System.out.println("Index " + i + ": " + arr[i]);
        }

        // Accept records from the user
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter 5 integers to update the array:");
        for (int i = 0; i < arr.length; i++) {
            arr[i] = scanner.nextInt();
        }

        // Print the updated values
        System.out.println("Updated values in the array:");
        for (int i = 0; i < arr.length; i++) {
            System.out.println("Index " + i + ": " + arr[i]);
        }

        scanner.close();
    }
}
```



```
<terminated> Program (9) [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v2024
Default values in the array:
Index 0: 0
Index 1: 0
Index 2: 0
Index 3: 0
Index 4: 0
Enter 5 integers to update the array:
10
20
30
40
50
Updated values in the array:
Index 0: 10
Index 1: 20
Index 2: 30
Index 3: 40
Index 4: 50
```

2. Declare a single-dimensional array of 5 integers inside the `main` method. Define a method named `acceptRecord` to get input from the terminal into the array and another method named `printRecord` to print the state of the array to the terminal.

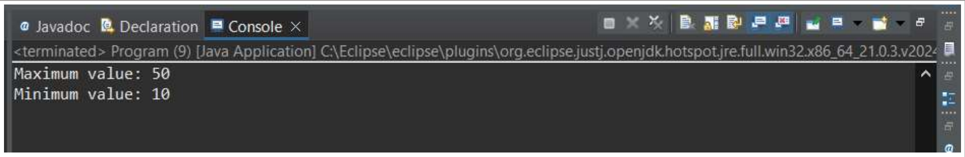
```java
package org.example.q1;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        // Declare a single-dimensional array of 5 integers
        int[] arr = new int[5];
        acceptRecord (arr);
        printRecord (arr);

    }
    Scanner sc = new Scanner (System.in);

        public static void acceptRecord (int [] arr) {
             Scanner sc = new Scanner (System.in);
             System.out.println("Enter 5 integers to update the array: ");
             for (int i =0; i<arr.length; i++) {
                    arr[i] = sc.nextInt ();
             }
        }
        public static void printRecord (int [] arr) {
             System.out.println("Array Contents : ");
             for (int i = 0; i < arr.length; i++) {
                System.out.println("Index " + i + ": " + arr[i]);
             }
        }
}
```



```
<terminated> Program (9) [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v2024
Enter 5 integers to update the array:
10
20
30
40
50
Array Contents :
Index 0: 10
Index 1: 20
Index 2: 30
Index 3: 40
Index 4: 50
```

3. Write a program to find the maximum and minimum values in a single-dimensional array of integers.

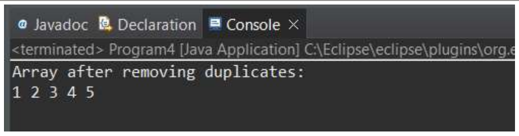
```
package org.example.q1;
import java.util.Scanner;

public class Program3 {

    public static void main(String[] args) {
        // Declare a single-dimensional array of 5 integers
        int[] arr = new int [] {10, 20, 30, 40, 50};
        int max = findMax (arr);
        int min = findMin (arr);

        System.out.println("Maximum value: " + max);
        System.out.println("Minimum value: " + min);
    }

        public static int findMax(int[] arr) {
            int max = arr[0];
            for (int i = 1; i < arr.length; i++) {
                if (arr[i] > max) {
                    max = arr[i];
                }
            }
            return max;
        }

        public static int findMin(int[] arr) {
            int min = arr[0];
            for (int i = 1; i < arr.length; i++) {
                if (arr[i] < min) {
                    min = arr[i];
                }
            }
            return min;

    }
}
```



```
<terminated> Program (9) [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v2024
Maximum value: 50
Minimum value: 10
```

4. Write a program to remove duplicate elements from a single-dimensional array of integers.

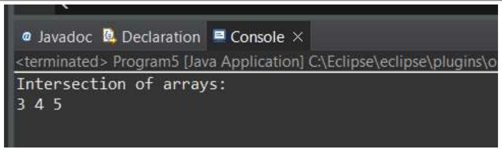
```
package org.example.q1;
import java.util.Set;
import java.util.HashSet;

public class Program4 {
       public static void main(String[] args) {
        int[] arr = {1, 2, 3, 2, 4, 5, 1};
        int[] uniqueArr = removeDuplicates(arr);

        System.out.println("Array after removing duplicates:");
        for (int num : uniqueArr) {
            System.out.print(num + " ");
        }
    }
    public static int[] removeDuplicates(int[] arr) {
        Set<Integer> set = new HashSet<>();
        for (int num : arr) {
            set.add(num);
        }
        return set.stream().mapToInt(Integer::intValue).toArray();
    }
}
```



```
<terminated> Program4 [Java Application] C:\Eclipse\eclipse\plugins\org.e
Array after removing duplicates:
1 2 3 4 5
```

5. Write a program to find the intersection of two single-dimensional arrays.

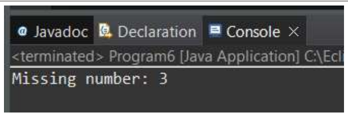
```
package org.example.q1;
import java.util.HashSet;
import java.util.Set;


public class Program5 {
      public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        int[] arr2 = {3, 4, 5, 6, 7};
        int[] intersection = findIntersection(arr1, arr2);

        System.out.println("Intersection of arrays:");
        for (int num : intersection) {
            System.out.print(num + " ");
        }
    }

    public static int[] findIntersection(int[] arr1, int[] arr2) {
        Set<Integer> set1 = new HashSet<>();
        for (int num : arr1) {
            set1.add(num);
        }

        Set<Integer> intersection = new HashSet<>();
        for (int num : arr2) {
            if (set1.contains(num)) {
                intersection.add(num);
            }
        }

        return intersection.stream().mapToInt(Integer::intValue).toArray();
    }
}
```



```
<terminated> Program5 [Java Application] C:\Eclipse\eclipse\plugins\o
Intersection of arrays:
3 4 5
```

6. Write a program to find the missing number in an array of integers ranging from 1 to N.

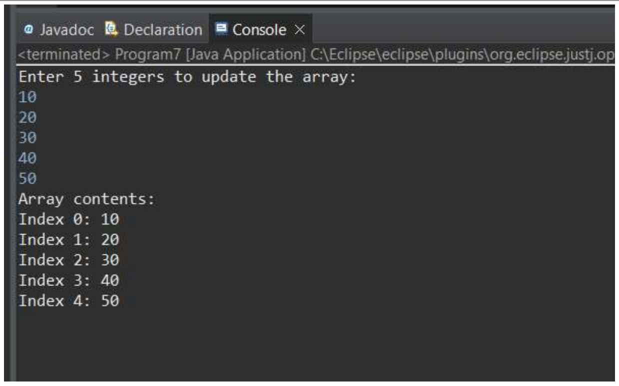
```
package org.example.q1;

public class Program6 {
      public static void main(String[] args) {
        int[] arr = {1, 2, 4, 5}; // Assuming numbers from 1 to 5
        int n = 5;
        int missingNumber = findMissingNumber(arr, n);

        System.out.println("Missing number: " + missingNumber);
    }

    public static int findMissingNumber(int[] arr, int n) {
        int totalSum = n * (n + 1) / 2;
        int arraySum = 0;
        for (int num : arr) {
            arraySum += num;
        }
        return totalSum - arraySum;
    }
}
```



```
<terminated> Program6 [Java Application] C:\Ecl
Missing number: 3
```

7. Declare a single-dimensional array as a field inside a class and instantiate it inside the class constructor. Define methods named `acceptRecord` and `printRecord` within the class and test their functionality.

```
package org.example.q1;
import java.util.Scanner;

public class Program7 {
	private int[] arr;

    // Constructor to initialize the array
    public Program7 (int size) {
        arr = new int[size];
    }

    // Method to accept input from the user
    public void acceptRecord() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter " + arr.length + " integers to update the array:");
        for (int i = 0; i < arr.length; i++) {
            arr[i] = scanner.nextInt();
        }
    }

    // Method to print the array
    public void printRecord() {
        System.out.println("Array contents:");
        for (int i = 0; i < arr.length; i++) {
            System.out.println("Index " + i + ": " + arr[i]);
        }
    }

    public static void main(String[] args) {
        Program7 manager = new Program7 (5);
        manager.acceptRecord();
        manager.printRecord();
    }
}
```

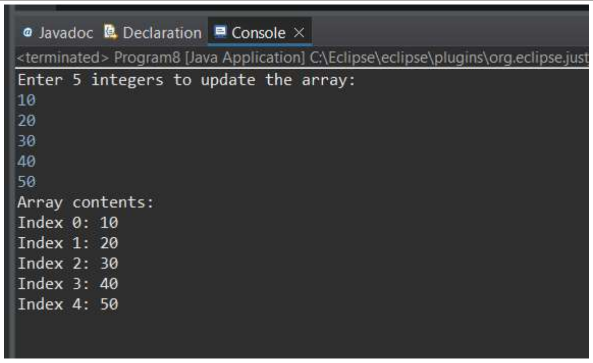


```
<terminated> Program7 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.justj.op
Enter 5 integers to update the array:
10
20
30
40
50
Array contents:
Index 0: 10
Index 1: 20
Index 2: 30
Index 3: 40
Index 4: 50
```

8. Modify the previous assignment to use getter and setter methods instead of `acceptRecord` and `printRecord`.

```
package org.example.q1;
import java.util.Scanner;

public class Program8 {
	private int[] arr;

	    // Constructor to initialize the array
	    public Program8(int size) {
	        arr = new int[size];
	    }

	    // Getter for the array
	    public int[] getArray() {
	        return arr;
	    }

	    // Setter for the array
	    public void setArray(int[] arr) {
	        this.arr = arr;
	    }

	    // Method to print the array
	    public void printRecord() {
	        System.out.println("Array contents:");
	        for (int i = 0; i < arr.length; i++) {
	            System.out.println("Index " + i + ": " + arr[i]);
	        }
	    }

	    public static void main(String[] args) {
	        Program8 manager = new Program8(5);
	        Scanner scanner = new Scanner(System.in);
	        int[] newArray = new int[5];
	        System.out.println("Enter 5 integers to update the array:");
	        for (int i = 0; i < newArray.length; i++) {
	            newArray[i] = scanner.nextInt();
	        }
	        manager.setArray(newArray);
```

```
            manager.printRecord();
            scanner.close();
        }
}
```



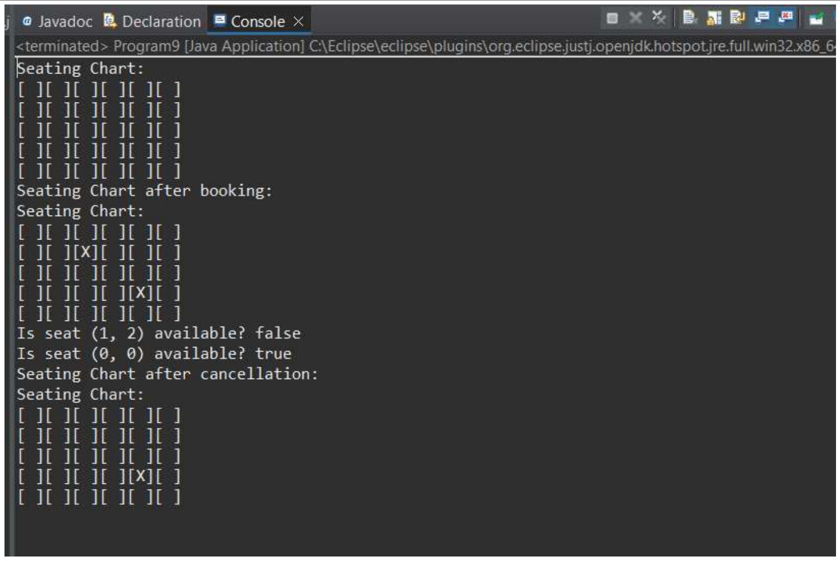
Javadoc Declaration Console

```
<terminated> Program8 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.just
Enter 5 integers to update the array:
10
20
30
40
50
Array contents:
Index 0: 10
Index 1: 20
Index 2: 30
Index 3: 40
Index 4: 50
```

9. You need to implement a system to manage airplane seat assignments. The airplane has seats arranged in rows and columns. Implement functionalities to:
   - Initialize the seating arrangement with a given number of rows and columns.
   - Book a seat to mark it as occupied.
   - Cancel a booking to mark a seat as available.
   - Check seat availability to determine if a specific seat is available.
   - Display the current seating chart.

```
package org.example.q1;

public class Program9 {
        private boolean[][] seats;

           // Initialize the seating arrangement with a given number of rows and columns
           public Program9(int rows, int cols) {
               seats = new boolean[rows][cols];
           }

           // Book a seat to mark it as occupied
           public boolean bookSeat(int row, int col) {
               if (isValidSeat(row, col)) {
                   if (!seats[row][col]) {
                       seats[row][col] = true;
                       return true; // Booking successful
                   } else {
                       System.out.println("Seat is already occupied.");
                       return false; // Booking failed
                   }
               } else {
                   System.out.println("Invalid seat number.");
                   return false; // Booking failed
               }
           }

           // Cancel a booking to mark a seat as available
           public boolean cancelBooking(int row, int col) {
               if (isValidSeat(row, col)) {
                   if (seats[row][col]) {
                       seats[row][col] = false;
                       return true; // Cancellation successful
                   } else {
                       System.out.println("Seat is not occupied.");
                       return false; // Cancellation failed
                   }
               } else {
                   System.out.println("Invalid seat number.");
                   return false; // Cancellation failed
               }
           }

           // Check seat availability to determine if a specific seat is available
           public boolean isSeatAvailable(int row, int col) {
               if (isValidSeat(row, col)) {
                   return !seats[row][col];
               } else {
                   System.out.println("Invalid seat number.");
                   return false; // Seat is not available
               }
           }

           // Display the current seating chart
           public void displaySeatingChart() {
               System.out.println("Seating Chart:");
               for (int i = 0; i < seats.length; i++) {
                   for (int j = 0; j < seats[i].length; j++) {
                       System.out.print(seats[i][j] ? "[X]" : "[ ]");
                   }
                   System.out.println();
               }
           }
```

```java
    // Check if the seat number is valid
    private boolean isValidSeat(int row, int col) {
        return row >= 0 && row < seats.length && col >= 0 && col < seats[row].length;
    }

    public static void main(String[] args) {
      Program9 airplane = new Program9(5, 6); // Example with 5 rows and 6 columns

        // Display the initial seating chart
        airplane.displaySeatingChart();

        // Book some seats
        airplane.bookSeat(1, 2);
        airplane.bookSeat(3, 4);

        // Display the seating chart after booking
        System.out.println("Seating Chart after booking:");
        airplane.displaySeatingChart();

        // Check seat availability
        System.out.println("Is seat (1, 2) available? " + airplane.isSeatAvailable(1, 2));
        System.out.println("Is seat (0, 0) available? " + airplane.isSeatAvailable(0, 0));

        // Cancel a booking
        airplane.cancelBooking(1, 2);

        // Display the seating chart after cancellation
        System.out.println("Seating Chart after cancellation:");
        airplane.displaySeatingChart();
    }
}
```



```
<terminated> Program9 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_6
Seating Chart:
[ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ]
Seating Chart after booking:
Seating Chart:
[ ][ ][ ][ ][ ][ ]
[ ][ ][X][ ][ ][ ]
[ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][X][ ]
[ ][ ][ ][ ][ ][ ]
Is seat (1, 2) available? false
Is seat (0, 0) available? true
Seating Chart after cancellation:
Seating Chart:
[ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][ ][ ]
[ ][ ][ ][ ][X][ ]
[ ][ ][ ][ ][ ][ ]
```