ASSIGNMENT NO.6

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.

Ans. package org.example.question1;

```
import java.util.Scanner;

public class Program1 {

public static void main(String[] args) {

		int[] arr = new int[5];

		// to traverse the default array

		for(int i = 0 ;i<arr.length; i++) {

			System.out.println(arr[i]);

		}

		// traversing the array by user input

		Scanner sc = new Scanner(System.in);

		for(int i= 0; i<arr.length; i++) {

			System.out.println("index " +i + " : ");

			arr[i] = sc.nextInt();

		}

		//printing the updated values

		for(int i= 0; i<arr.length; i++) {

			System.out.println("index "+ i+ " : " + arr[i]);

		}

		sc.close();

	}

}
```

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.

Ans.

```
package org.example.question2;

import java.util.Scanner;

public class Program2 {

	public static void acceptRecord(int[] arr) {

		Scanner sc = new Scanner(System.in);

		for(int i= 0; i<arr.length; i++) {

			System.out.println("index " + i + " : ");

			arr[i] = sc.nextInt();

		}

		sc.close();

	}

	public static void printRecord(int[] arr) {

		for(int i= 0; i<arr.length; i++) {

			System.out.println("index "+ i+ " : " +arr[i]);

		}

	}

	public static void main(String[] args) {

		int[] arr = new int[5];

		Program2.acceptRecord(arr);

		Program2.printRecord(arr);

	}

}
```

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

Ans.
```
package org.example.question3;

import java.util.Scanner;

public class Program3 {

	public static void accepRecord(int[] arr) {

		Scanner sc = new Scanner(System.in);


		for(int i= 0; i<arr.length; i++) {

			System.out.println("index "+ i + " : ");

			arr[i] = sc.nextInt();

		}

	}


	public static void printRecord(int[] arr) {

		for(int i= 0; i<arr.length; i++) {

			System.out.println("index "+ i+ " : " +arr[i]);

		}

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

		Scanner sc = new Scanner(System.in);

		System.out.println("Enter the number of elements you want to put in array");

		int n = sc.nextInt();


		int[] arr  = new int[n];
```

```
		Program3.accepRecord(arr);
		Program3.printRecord(arr);
		Program3.max(arr);
		Program3.min(arr);
	}
	private static void max(int[] arr) {
		int maxi = arr[0];
		for(int  i = 0; i<arr.length; i++) {
			if(arr[i]>maxi) {
				maxi = arr[i];
			}
		}
		System.out.println("The max number is : "+ maxi);
	}
	private static void min(int[] arr) {
		int mini = arr[0];
		for(int  i = 1; i<arr.length; i++) {
			if(arr[i]<mini) {
				mini = arr[i];
			}
		}
		System.out.println("The mini number is : "+ mini);
	}
}
```

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

Ans.

```
package org.example.question4;

import java.util.Arrays;

public class Program4 {


    private static int[] removeDuplicates(int[] array) {

        int[] temp = new int[array.length];

    int size = 0;


    // Iterate over each element in the original array

    for (int i = 0; i < array.length; i++) {

        boolean isDuplicate = false;


        // Check if the element is already in the temp array

        for (int j = 0; j < size; j++) {

            if (array[i] == temp[j]) {

                isDuplicate = true;

                break;

            }

        }


        // If the element is not a duplicate, add it to the temp array

        if (!isDuplicate) {

            temp[size++] = array[i];
```

```
            }
        }

        // Create a new array with the exact size of unique elements
        int[] uniqueArray = new int[size];
        System.arraycopy(temp, 0, uniqueArray, 0, size);

        return uniqueArray;
    }
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 2, 4, 5, 5, 6};
    int[] result = removeDuplicates(array);

    System.out.println("Array with duplicates removed: " + Arrays.toString(result));

    }
}
```

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

Ans. package org.example.question5;

```
import java.util.Arrays;

public class Program5 {
```

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

		int[] array1 = {1,2,3,4,5};

		int[] array2 = {3,4,5,6,7};


		int[] intersection = findIntersection(array1, array2);


		System.out.println("Intersection of the two arrays: "
+Arrays.toString(intersection));

	}


	private static int[] findIntersection(int[] array1, int[] array2) {

		int [] temp = new int[array1.length]; // array to store intersection elements

		int size = 0;



		for(int i= 0; i<array1.length; i++) {

			for(int j= 0; j<array2.length; j++) {

				if(array1[i]== array2[j]) {

					boolean alreadyExists = false;

					for(int k= 0; k<size; k++) {

						if(temp[k]==array1[i]) {

						alreadyExists = true;

						break;

						}
```

```
                                        }


                                        if(!alreadyExists) {
                                                temp[size++] = array1[i];
                                        }
                                        break;
                                }

                        }
                }

                int[] result = new int[size];
                for(int i= 0; i<size; i++) {
                        result[i] = temp[i];
                }
                return result;
        }

}
```

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

Ans. package org.example.question6;

```
public class Program6 {
```

```
    public static void main(String[] args) {
        int arr[] = {1,2,4,5,6};
        int missingNumber = findMissingNumber(arr, 6);
        System.out.println("the missing number is : " +missingNumber);

    }

    private static int findMissingNumber(int[] arr, int N) {
        int expectedSum = N*(N+1)/2;
        int actualSum = 0;

        for(int num: arr) {
            actualSum+= num;
        }

        return expectedSum - actualSum;
    }
}
```

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.

Ans. package org.example.question7;

```
public class Program7 {

    int[] array;

    public Program7(int size) {
        array = new int [5];
    }

    private void printRecord(int[] newRecords) {

        System.out.print("Records: ");
for (int record : array) {
    System.out.print(record + " ");
}
System.out.println();
    }

    private void acceptRecord(int[] newRecords) {
        if (newRecords.length <= array.length) {
        for (int i = 0; i < newRecords.length; i++) {
            array[i] = newRecords[i];
        }
      } else {
        System.out.println("Error: Input array size exceeds the initialized array size.");
      }
```

```
    }

    public static void main(String[] args) {
        Program7 prog = new Program7(5);

        int[] newRecords = {10,20,30,40,50};

        prog.acceptRecord(newRecords);
        prog.printRecord(newRecords);

    }
}
```

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

Ans. package org.example.question8;

```
import org.example.question8.Program8;

public class Program8 {

    int[] array;

    public Program8(int size) {
```

```
            array = new int [5];
        }

        public int[] getRecords() {
    return array;
}
        public void setRecords(int[] newRecords) {
    // Ensure newRecords length does not exceed the size of the records array
    if (newRecords.length <= array.length) {
        for (int i = 0; i < newRecords.length; i++) {
            array[i] = newRecords[i];
        }
    } else {
        System.out.println("Error: Input array size exceeds the initialized array size.");
    }
}

        public static void main(String[] args) {
            Program8 prog = new Program8(5);

            int[] newRecords = {10,20,30,40,50};

            prog.setRecords(newRecords);

             int[] records = prog.getRecords();
```

```
            System.out.print("Records: ");

            for (int record : records) {

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

            }

            System.out.println();


        }

}
```

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.

Ans.

```
package org.example.question9;

public class AirplaneSeating {

        private boolean[][] seats; // 2D array to represent seat availability

          // Constructor to initialize the seating arrangement
          public AirplaneSeating(int rows, int columns) {
          seats = new boolean[rows][columns]; // False means available, True means
occupied
          }

          // Method to book a seat
          public boolean bookSeat(int row, int column) {
             // Check if the seat is within bounds
             if (isValidSeat(row, column)) {
                if (!seats[row][column]) { // Check if the seat is available
                   seats[row][column] = true; // Mark seat as occupied
                   return true; // Booking successful
                } else {
```

```
            System.out.println("Seat already booked.");
            return false; // Booking failed
        }
    } else {
        System.out.println("Invalid seat position.");
        return false; // Booking failed
    }
}

// Method to cancel a booking
public boolean cancelBooking(int row, int column) {
    // Check if the seat is within bounds
    if (isValidSeat(row, column)) {
        if (seats[row][column]) { // Check if the seat is occupied
            seats[row][column] = false; // Mark seat as available
            return true; // Cancellation successful
        } else {
            System.out.println("Seat is not booked.");
            return false; // Cancellation failed
        }
    } else {
        System.out.println("Invalid seat position.");
        return false; // Cancellation failed
    }
}

// Method to check seat availability
public boolean isSeatAvailable(int row, int column) {
    // Check if the seat is within bounds
    if (isValidSeat(row, column)) {
        return !seats[row][column]; // Return true if the seat is available
    } else {
        System.out.println("Invalid seat position.");
        return false; // Seat is considered unavailable if invalid
    }
}

// Method to display the current seating chart
public void displaySeatingChart() {
    System.out.println("Current Seating Chart:");
    for (int i = 0; i < seats.length; i++) {
        for (int j = 0; j < seats[i].length; j++) {
            if (seats[i][j]) {
                System.out.print(" X "); // X represents an occupied seat
            } else {
                System.out.print(" O "); // O represents an available seat
            }
```

```
                }
                System.out.println(); // Newline for the next row
            }
        }

        // Helper method to check if the seat position is valid
        private boolean isValidSeat(int row, int column) {
            return row >= 0 && row < seats.length && column >= 0 && column <
seats[0].length;
        }

    public static void main(String[] args) {
            AirplaneSeating seating = new AirplaneSeating(5, 6);

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

    // Book some seats
    seating.bookSeat(2, 3);
    seating.bookSeat(0, 0);
    seating.bookSeat(4, 5);

    // Display the seating chart after booking
    seating.displaySeatingChart();

    // Check seat availability
    System.out.println("Seat (2, 3) available: " + seating.isSeatAvailable(2, 3));
    System.out.println("Seat (1, 1) available: " + seating.isSeatAvailable(1, 1));

    // Cancel a booking
    seating.cancelBooking(2, 3);

    // Display the seating chart after cancellation
    seating.displaySeatingChart();
  }
      }
```