LA GRANDEE INTERNATIONAL COLLGEE.

Data Structure and Algorithm.
Assignment - I

Submitted To:
Er. Ashwin. Poudel.

Name: Sachin Timilsina.

BCA 3rd Sem, (2025).

DSA

1. Write a program to reverse an array.

Algorithm:

```
1) Start.
2) Input an array of size n.
3) Initialize two pointers.
        start = 0
        end = n-1
4) Repeat the following step while start < end
        a) Swap the element at start and end.
        b) Increment start by 1.
        c) Decrement end by 1.
5) End loop when start ≥ end.
6) The array is reversed.
7) Display array.
8) Stop.
```

Time Complexity : $O(n)$
Space Complexity : $O(1)$

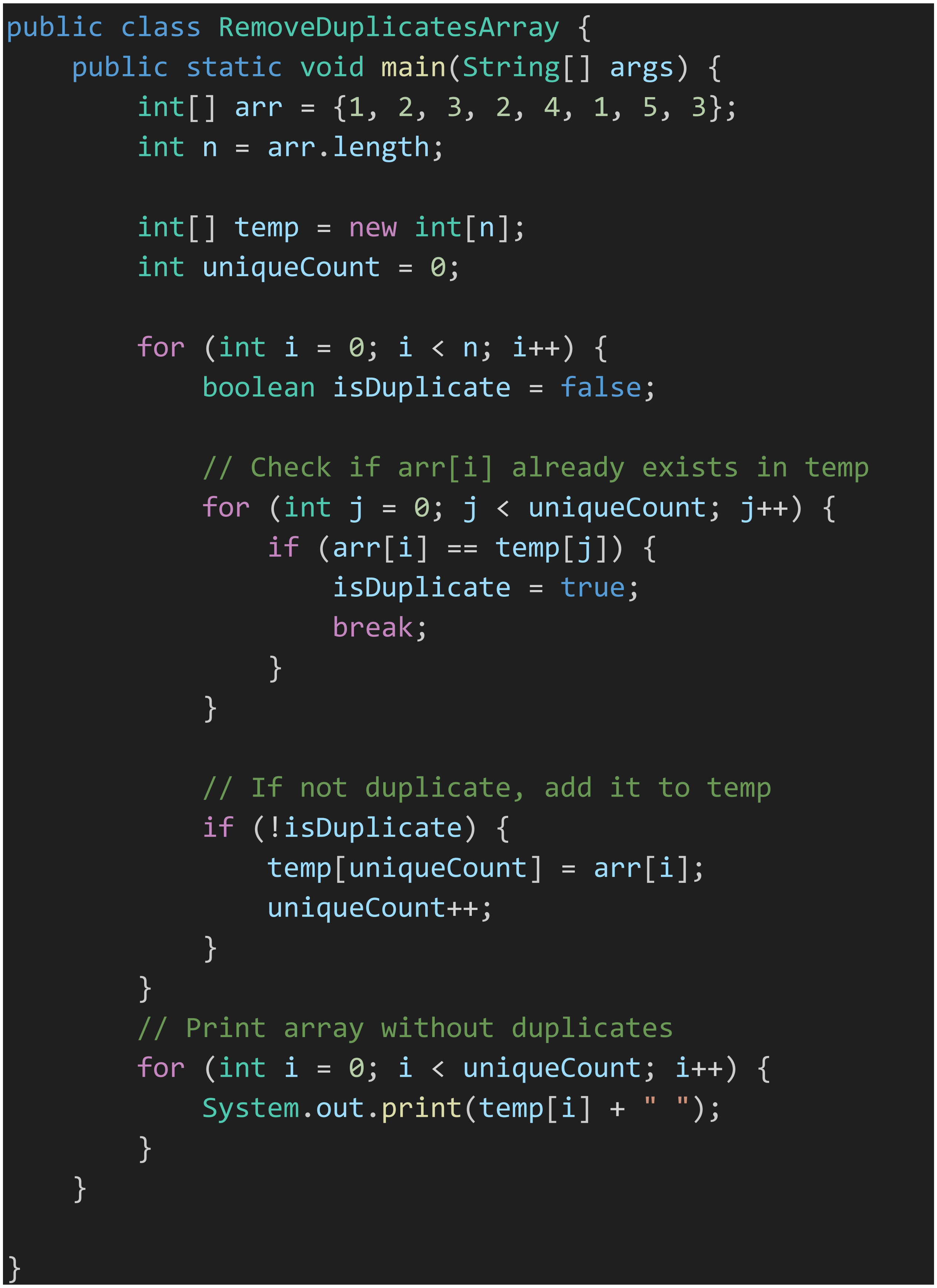
```java
public class ReverseArray {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};

        // Reverse the array
        int start = 0;
        int end = arr.length - 1;
        while (start < end) {
            // Swap elements
            int temp = arr[start];
            arr[start] = arr[end];
            arr[end] = temp;

            start++;
            end--;
        }

        System.out.println("\nReversed Array:");
        for (int num : arr) {
            System.out.print(num + " ");
        }
    }
}
```

2. Write a program to remove duplicate from array

Algorithm:

```
1) Start.
2) Input an array of size n.
3) Create an ~~array~~ empty temporary array called
   uniqueArray.
4) For each element x in original array:
     a) Check if x is already present in uniqueArray.
     b) If not present, add x to UniqueArray.
5) After loop, uniqueArray will contain only unique
   elements.
6) Display elements of uniqueArray.
7) Stop.
```

Time Complexity : $O(n^2)$
Space Complexity : $O(n)$.



```java
public class RemoveDuplicatesArray {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 2, 4, 1, 5, 3};
        int n = arr.length;

        int[] temp = new int[n];
        int uniqueCount = 0;

        for (int i = 0; i < n; i++) {
            boolean isDuplicate = false;

            // Check if arr[i] already exists in temp
            for (int j = 0; j < uniqueCount; j++) {
                if (arr[i] == temp[j]) {
                    isDuplicate = true;
                    break;
                }
            }

            // If not duplicate, add it to temp
            if (!isDuplicate) {
                temp[uniqueCount] = arr[i];
                uniqueCount++;
            }
        }
        // Print array without duplicates
        for (int i = 0; i < uniqueCount; i++) {
            System.out.print(temp[i] + " ");
        }
    }

}
```

3) Write a program to print the numbers from an Array.

Algorithm:

```
1) Start.
2) Input an array of size n.
3) Set a loop counter i=0.
4) Repeat the following step while i < n:
        a) Print the element in position i.
        b) Increment i by 1
5) End when ~~i>=n~~ i==n.
6) Stop.
```

Time Complexity: $O(n)$
Space Complexity: $O(1)$

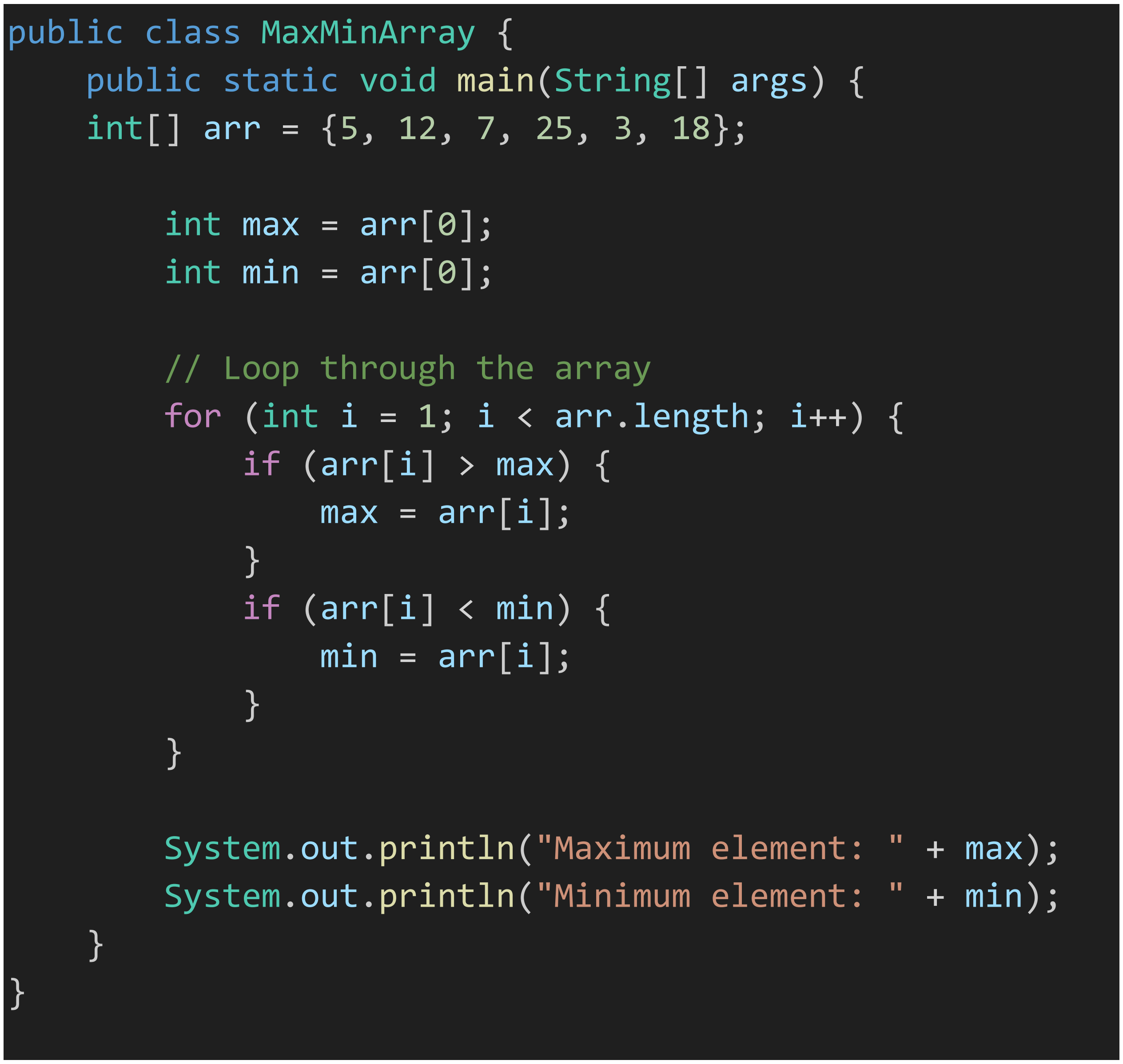
```java
public class PrintArray {
    public static void main(String[] args) {
       int[] arr = {10, 20, 30, 40, 50};

       for (int i = 0; i < arr.length; i++) {
           System.out.println(arr[i]);
        }
    }
}
```

4). Write a program to find max & min element in an array.

Algorithm:

```
1) Start.
2) Input an array of size n.
3) Initialize two variables.
       max = First element of array.
       min = First element of array.
4) Set i=1 ( First element is already considered).
5) Repeat the following step while i<n:
       a) If element at position 'i' is greater than max,
          update max with that element.
       b) If element at position 'i' is smaller than
          min, update min with that element.
6) End loop after checking all elements.
7) Display max & min.
8) Stop.
```

Time Complexity : O(n)
Space Complexity : O(1)



```java
public class MaxMinArray {
    public static void main(String[] args) {
    int[] arr = {5, 12, 7, 25, 3, 18};

        int max = arr[0];
        int min = arr[0];

        // Loop through the array
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
            if (arr[i] < min) {
                min = arr[i];
            }
        }

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

5. Write a program to implement GCD of two numbers using recursion.

Algorithm :

```
1) Start
2) Input two number : a and b.
3) Check if b is equal to 0.
    a) If yes, then GCD = a.
    b) If no, then recursively call the same function
       with (b, a % b).
4) Continue this until b == 0.
5) Return value of a as GCD.
6). Display GCD.
7) Stop.
```

Time Complexity: $O(\log(\min(a,b)))$
Space Complexity: $O(\log(\min(a,b)))$

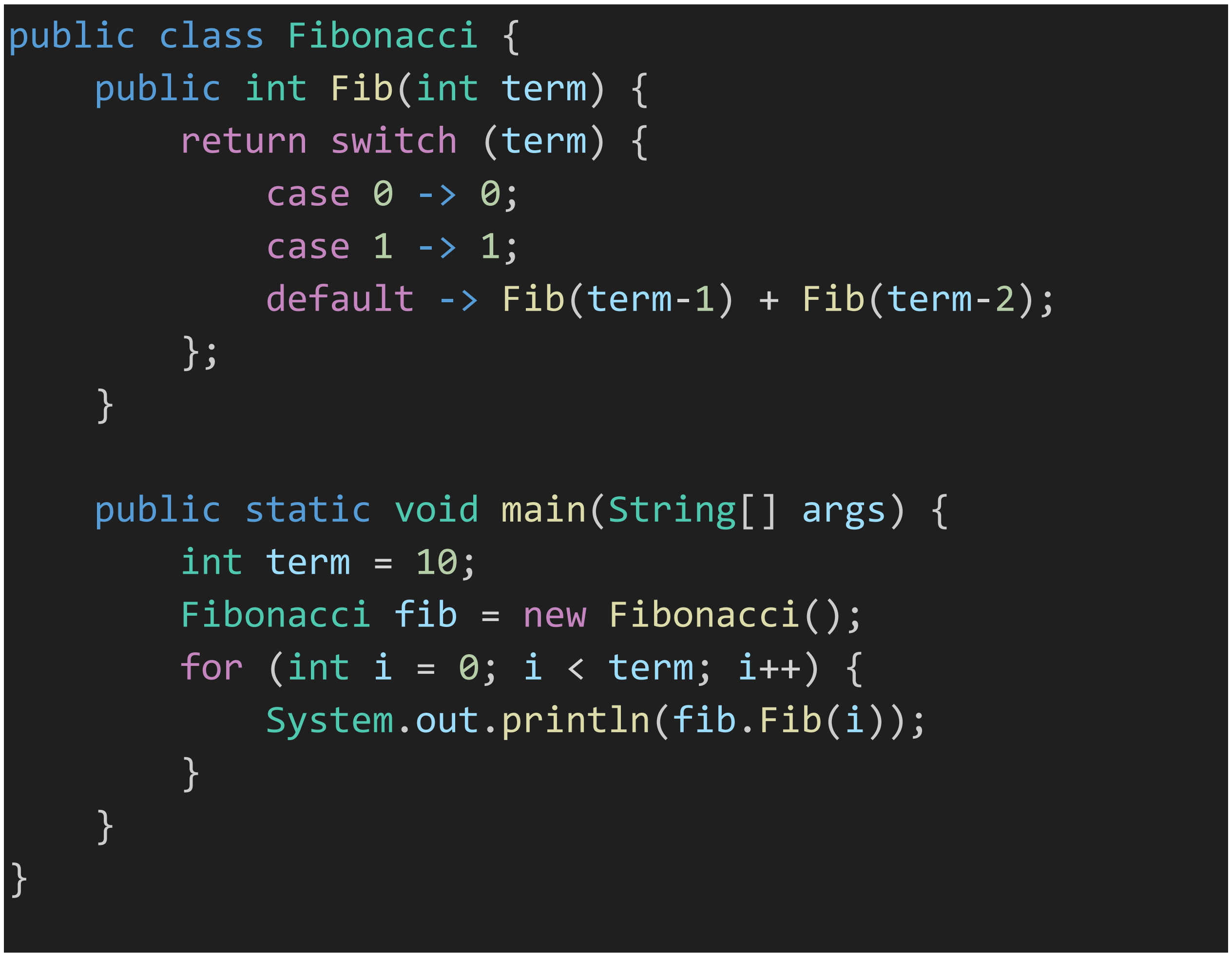
```java
public class GCDRecursion {
    // Recursive method to find GCD
    static int gcd(int a, int b) {
        if (b == 0) return a;
        else return gcd(b, a % b);
    }

    public static void main(String[] args) {
        int num1 = 48;
        int num2 = 18;

        int result = gcd(num1, num2);

        System.out.println("GCD: " + result);
    }

}
```

6. Write a program to implement fibonacci Series.

Algorithm:

```
1) Start.
2) Input n (number of terms).
3) Define a recursive function fibonacci(num)
   that:
        a) Return 0 if n==0.
        b) Return 1 if n==1.
        c) Otherwise return fibonacci(num-1)+
              fibonacci(num-2).
4) In the main program, use a loop from 0 to
   n-1.
5)      a) For each iteration call fibonacci(i), print result.
5) End loop after printing all n terms.
6) Stop.
```

Time Complexity: $O(2^n)$
Space Complexity: $O(n)$.



```java
public class Fibonacci {
    public int Fib(int term) {
        return switch (term) {
            case 0 -> 0;
            case 1 -> 1;
            default -> Fib(term-1) + Fib(term-2);
        };
    }

    public static void main(String[] args) {
        int term = 10;
        Fibonacci fib = new Fibonacci();
        for (int i = 0; i < term; i++) {
            System.out.println(fib.Fib(i));
        }
    }
}
```

Github Repository: https://www.github.com/Sachin-Timilsina/DSA-Assignments.git