1. Anagram Program

```java
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class AnagramProgram {
    public static boolean areAnagrams(String s1, String s2) {
        Map<Character, Integer> freq1 = new HashMap<>();
        Map<Character, Integer> freq2 = new HashMap<>();
        for (char letter : s1.toCharArray()) {
            freq1.put(letter, freq1.getOrDefault(letter, 0) + 1);
        }
        for (char letter : s2.toCharArray()) {
            freq2.put(letter, freq2.getOrDefault(letter, 0) + 1);
        }
        return freq1.equals(freq2);
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter first string: ");
        String s1 = scan.next();
        System.out.println("Enter second string: ");
        String s2 = scan.next();
        System.out.println(areAnagrams(s1, s2));
        scan.close();
    }
}
```

```
Enter first string:
venkat
Enter second string:
kenvat
true
PS D:\Professional\Programming\12 Nov 2024>
```

2. Longest Palindrome

```java
import java.util.Scanner;

public class LongestPalindrome {
    public static String longestPalindrome(String s) {
        String result = Character.toString(s.charAt(0));
        int left = 0, right = s.length() - 1;
        while (right - left > 0) {
            int l = left, r = right;
            boolean check = true;
            int mid = (int) (r - l) / 2;
            for (int i = 0; i <= mid; i++) {
                if (s.charAt(left + i) != s.charAt(right - i)) {
                    check = false;
                    break;
                }
            }
            if (check)
                break;
            if (right < s.length() - 1) {
                left++;
                right++;
            } else {
                int diff = right - left;
                left = 0;
                right = left + diff - 1;
            }
        }
        if (right - left > 0) {
            result = s.substring(left, right + 1);
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter a string: ");
        String input = scan.nextLine();
        System.out.println(longestPalindrome(input));
        scan.close();
    }
}
```

```
Enter a string:
aaaabbaa
The longest palindrome is: aabbaa
```

3. Longest Consecutive Subsequence

```java
import java.util.Arrays;
import java.util.Scanner;

public class LongestSubsequence {
    public static int findLongestConseqSubseq(int[] arr) {
        Arrays.sort(arr);
        int count = 1;
        int max = 1;
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] - arr[i - 1] == 1) {
                count++;
            } else if (arr[i] - arr[i - 1] == 0) {
                continue;
            } else {
                count = 1;
            }
            max = Math.max(max, count);
        }
        return max;
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int length = scan.nextInt();
        int[] arr = new int[length];
        for (int i = 0; i < length; i++) {
            arr[i] = scan.nextInt();
        }
        System.out.println(findLongestConseqSubseq(arr));
        scan.close();
    }
}
```

```
9
1 6 3 10 5 2 9 4 13
Result: 6
```

4. Rat in a Maze (Using Backtracking)

```java
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

public class RatInMaze {
    private static Set<String> routes = new HashSet<>();

    private static void backtracking(
        int[][] mat, int x, int y, String route, int[][] visited
    ) {
        if (mat[x][y] == 0) {
            return;
        } else if (x == mat.length - 1 && y == mat.length - 1) {
            routes.add(route);
        } else {
            visited[x][y] = 1;
            // Move Left
            if (y - 1 >= 0 && visited[x][y - 1] == 0
                && mat[x][y - 1] == 1) {
                String newRoute = route + "L";
                backtracking(mat, x, y - 1, newRoute, visited);
            }
            // Move Right
            if (y + 1 < mat.length && visited[x][y + 1] == 0
                && mat[x][y + 1] == 1) {
                String newRoute = route + "R";
                backtracking(mat, x, y + 1, newRoute, visited);
            }
            // Move Up
            if (x - 1 >= 0 && visited[x - 1][y] == 0
                && mat[x - 1][y] == 1) {
                String newRoute = route + "U";
                backtracking(mat, x - 1, y, newRoute, visited);
            }
            // Move Down
            if (x + 1 < mat.length && visited[x + 1][y] == 0
                && mat[x + 1][y] == 1) {
                String newRoute = route + "D";
                backtracking(mat, x + 1, y, newRoute, visited);
            }
            visited[x][y] = 0;
        }
    }
```

```
    private static ArrayList<String> findPath(int[][] mat) {
        routes.clear();
        int[][] visited = new int[mat.length][mat.length];
        backtracking(mat, 0, 0, "", visited);
        ArrayList<String> result = new ArrayList<>(routes);
        return result;
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int length = scan.nextInt();
        int[][] matrix = new int[length][length];
        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                matrix[i][j] = scan.nextInt();
            }
        }
        ArrayList<String> result = findPath(matrix);
        System.out.println((result.isEmpty()) ? "-1" : result);
        scan.close();
    }
}
```

```
4
1 0 0 0
1 1 0 0
1 1 0 0
0 1 1 1
[DRDDRR, DDRDRR]
```

5. Row with Maximum Ones

```
import java.util.Scanner;

public class RowMaxOnes {
    public static int rowWithMax1s(int arr[][]) {
        int index = -1;
        int maxCount = 0;
        for (int i = 0; i < arr.length; i++) {
            int count = 0;
            for (int j = 0; j < arr[i].length; j++) {
```

```
                if (arr[i][j] == 1) {
                    count++;
                }
            }
            if (count > maxCount) {
                maxCount = count;
                index = i;
            }
        }
        return index;
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int row = scan.nextInt();
        int col = scan.nextInt();
        int[][] arr = new int[row][col];
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                arr[i][j] = scan.nextInt();
            }
        }
        System.out.printf("Result: %d\n", rowWithMax1s(arr));
        scan.close();
    }
}
```

```
4 3
1 0 0
0 1 1
1 1 1
0 0 0
Result: 2
```