Array List

1. Beautiful Array-List

```java
import java.util.*;
public class BeautifulArrayList {
    public static void divideConquer (int start, int increment, ArrayList<Integer>res, int n) {
        if (start + increment > n) {
            res.add(start);
            return;
        }
        divideConquer(start, 2*increment, res, n);
        divideConquer(start + increment, 2*increment, res, n);
    }
    public static ArrayList <Integer> beautifulArray (int n) {
        ArrayList<Integer>res = new ArrayList<>();
        divideConquer (1, 1, res, n);
        return res;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        ArrayList<Integer>list = new ArrayList<>();
        System.out.print("Enter Size of ArrayList: ");
        int n = sc.nextInt();
        System.out.print("Beautiful ArrayList is ");
        System.out.println(beautifulArray(n));
    }
}
```

2. Container with Most Water

```java
import java.util.ArrayList;

public class ContainerWithMostWater {
                                    // 2 POINTER APPROACH (O(n))
    public static int storeWater(ArrayList<Integer>height) {
        int maxWater=0;
        int lp=0;   // left pointer
        int rp=height.size()-1; // right pointer
        while (lp < rp) {
            // Calculating water area
            int ht = Math.min(height.get(lp), height.get(rp));
            int width = rp - lp;
            int currWater = ht * width;
            maxWater = Math.max(maxWater, currWater);
            // Updating Pointer
            if (height.get(lp) < height.get(rp)) {
                lp++;
            } else {
                rp--;
            }
        }
        return maxWater;
    }
    public static void main(String[] args) {
        ArrayList<Integer>height = new ArrayList<>();
```

```
        // 1, 8, 6, 2, 5, 4, 8, 3, 7
        height.add(1);  height.add(8);  height.add(6);  height.add(2);  height.add(5);
        height.add(4);  height.add(8);  height.add(3);  height.add(7);
        System.out.println(storeWater(height));
    }
}
```

3. Container with Most Water 2nd Approach

```
import java.util.ArrayList;

public class ContainerWithMostWaterIInd {
                                        // BRUTE FORCE (O(n^2))
    public static int storeWater(ArrayList<Integer>height) {
        int maxWater = 0;
        for(int i = 0; i<height.size(); i++) {
            for(int j=i+1; j<height.size(); j++) {
                int ht = Math.min(height.get(i), height.get(j));
                int width = j-i;
                int currWater = ht * width;
                maxWater = Math.max(maxWater, currWater);
            }
        }
        return maxWater;
    }
    public static void main(String[] args) {
        ArrayList<Integer>height = new ArrayList<>();
        // 1, 8, 6, 2, 5, 4, 8, 3, 7
        height.add(1);  height.add(8);  height.add(6);  height.add(2);  height.add(5);
        height.add(4);  height.add(8);  height.add(3);  height.add(7);
        System.out.println(storeWater(height));
    }
}
```

4. Lonely Numbers

```
import java.util.*;
public class LonelyNums {
    public static ArrayList<Integer>findLonely (ArrayList<Integer>nums) {
        Collections.sort(nums);
        ArrayList<Integer>list = new ArrayList<>();
        for (int i = 1; i < nums.size()-1; i++) {
            if (nums.get(i-1) + 1 < nums.get(i) && nums.get(i) + 1 < nums.get(i+1)) {
                list.add(nums.get(i));
            }
        }
        if (nums.size() == 1) {
            list.add(nums.get(0));
        }
        if (nums.size() > 1) {
            if (nums.get(0) + 1 < nums.get(1)) {
                list.add(nums.get(0));
            }
            if (nums.get(nums.size()-2) + 1 < nums.get(nums.size()-1)) {
                list.add(nums.get(nums.size()-1));
            }
```

```
        }
        return list;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        ArrayList<Integer>list = new ArrayList<>();
        System.out.print("Enter no. of Elements in List: ");
        int n = sc.nextInt();
        System.out.println("Enter Elements: ");
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        System.out.print("Lonely Elements in ArrayList: ");
        System.out.println(findLonely(list));
    }
}
```

5. Maximum in an ArrayList

```
import java.util.ArrayList;

public class MaximumInAnAL {
    public static void main(String[] args) {
        ArrayList<Integer>list = new ArrayList<>();
        list.add(2);
        list.add(5);
        list.add(6);
        list.add(9);
        list.add(8);
        int max = Integer.MIN_VALUE;
        for (int i=0; i<list.size(); i++) {
            if (max < list.get(i)) {
                max = list.get(i);
            }
        }
        System.out.println("Max Element: " + max);
    }
}
```

6. Monotonic ArrayList

```
import java.util.*;
public class MonotonicAL {
    public static boolean isMonotonic (ArrayList<Integer>A) {
        boolean inc = true;
        boolean dec = true;
        for (int i = 0; i < A.size()-1; i++) {
            if (A.get(i) > A.get(i+1)) {
                inc = false;
            }
            if (A.get(i) < A.get(i+1)) {
                dec = false;
            }
        }
        return inc || dec;
    }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        ArrayList<Integer>list = new ArrayList<>();
        System.out.print("Enter no. of Elements in List: ");
        int n = sc.nextInt();
        System.out.println("Enter Elements: ");
        for (int i = 0; i<n; i++) {
            list.add(sc.nextInt());
        }
        System.out.print("Given ArrayList is Monotonic. - ");
        System.out.println(isMonotonic(list));
    }
}
```

7. Most Frequent Number

```
import java.util.*;
public class MostFrequentNumber {
    public static int mostFrequent (ArrayList<Integer>nums, int key) {
        int[] result = new int[1000];
        for (int i = 0; i < nums.size()-1; i++) {
            if (nums.get(i) == key) {
                result[nums.get(i+1)-1]++;
            }
        }
        int max = Integer.MIN_VALUE;
        int ans = 0;
        for (int i = 0; i < 1000; i++) {
            if (result[i] > max) {
                max = result[i];
                ans = i+1;
            }
        }
        return ans;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        ArrayList<Integer>list = new ArrayList<>();
        System.out.print("Enter no. of Elements in List: ");
        int n = sc.nextInt();
        System.out.print("Enter Elements: ");
        for (int i=0; i<n; i++) {
            list.add(sc.nextInt());
        }
        System.out.print("Enter Key: ");
        int key = sc.nextInt();
        System.out.print("Most Frequent Number Following Key is: ");
        System.out.println(mostFrequent(list, key));
    }
}
```

8. Multi-Dimensional ArrayList

```java
import java.util.ArrayList;

public class MultiDarrayList {
    public static void main(String[] args) {
        ArrayList<ArrayList<Integer>>mainList = new ArrayList<>();
        ArrayList<Integer>list1 = new ArrayList<>();
        ArrayList<Integer>list2 = new ArrayList<>();
        ArrayList<Integer>list3 = new ArrayList<>();
        for(int i=1; i<=5; i++) {
            list1.add(i*1); //1 2 3 4 5
            list2.add(i*2); //2 4 6 8 10
            list3.add(i*3); //3 6 9 12 15
        }
        mainList.add(list1);
        mainList.add(list2);
        mainList.add(list3);
        System.out.print("Multi-D ArrayList: ");
        System.out.println(mainList);
        // NESTED LOOPS
        for(int i=0; i<mainList.size(); i++){
            ArrayList<Integer>currList = mainList.get(i);
            for(int j=0; j<currList.size(); j++) {
                System.out.print(currList.get(j)+ " ");
            }
            System.out.println();
        }
    }
}
```

9. Operations on ArrayList

```java
import java.util.ArrayList;

public class OperationsOnAL {
    public static void main(String[] args) {
        ArrayList<Integer>list = new ArrayList<>();
        // ADD ELEMENT -- O(1)
        System.out.println("---Adding Element---");
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.add(5);
        System.out.println(list);
        // Adding at a specific index -- O(n)
        System.out.println("---Adding at Specific Index---");
        list.add(1,9);
        System.out.println(list);
        // GET ELEMENT -- O(1)
        System.out.println("---Get an Element---");
        int element = list.get(2);
        System.out.println(element);
        // DELETE ELEMENT -- O(n)
        System.out.println("---Delete an Element---");
        list.remove(1);
```

```
        System.out.println(list);
        // SET ELEMENT -- O(n)
        System.out.println("---Set an Element---");
        list.set(2, 10);
        System.out.println(list);
        // CONTAINS ELEMENT -- O(n)
        System.out.println("---Check Contains Element---");
        System.out.println(list.contains(1));
        System.out.println(list.contains(11));
    }
}
```

10. Pair Sum 1st Approach

```
import java.util.*;
                                // BRUTE FORCE - O(n^2)
public class PairSumIst {
    public static boolean pairSum1 (ArrayList<Integer>list, int target) {
        for (int i = 0; i < list.size(); i++) {
            for (int j = i+1; j < list.size(); j++) {
                if (list.get(i) + list.get(j) == target) {
                    return true;
                }
            }
        }
        return false;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        ArrayList<Integer>list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.add(5);
        list.add(6);
        // 1, 2, 3, 4, 5, 6
        System.out.print("Enter Target: ");
        int tar = sc.nextInt();
        System.out.println(pairSum1(list, tar));
    }
}
```

11. Pair Sum 2nd Approach

```
import java.util.*;
public class PairSumIInd {
    public static boolean pairSum2 (ArrayList<Integer>list, int target) {
        int bp = -1; // Breaking Point
        int n = list.size();
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i) > list.get(i+1)) {
                bp = i;
                break;
            }
        }
```

```
        int lp = bp + 1;    // smallest
        int rp = bp;    // largest
        while (lp != rp) {
            // Case 1
            if (list.get(rp) + list.get(lp) == target) {
                return true;
            }
            // Case 2 & 3
            if (list.get(lp) + list.get(rp) < target) {
                lp = (lp+1) % n;
            } else {
                rp = (n+rp-1) % n;
            }
        }
        return false;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        ArrayList<Integer>list = new ArrayList<>();
        System.out.print("Enter no. of elements in the List: ");
        int n = sc.nextInt();
        System.out.println("Enter Elements of List: ");
        for(int i=0; i<n; i++) {
            list.add(sc.nextInt());
        }
        System.out.print("Enter Target: ");
        int tar = sc.nextInt();
        System.out.print("Output: ");
        System.out.println(pairSum2(list, tar));
    }
}
```

12. Pair Sum 3rd Approach

```
import java.util.*;
public class PairSumIstsecondApproach {
                                        // 2 pointers approach
    public static boolean pairSum1 (ArrayList<Integer>list, int target) {
        int lp = 0; // left pointer
        int rp = list.size()-1; // right pointer
        while (lp != rp) {
            // Case 1
            if (list.get(lp) + list.get(rp) == target) {
                return true;
            }
            // Case 2
            if (list.get(lp) + list.get(rp) < target) {
                lp++;
            }
            // Case 3
            else {
                rp--;
            }
        }
        return false;
    }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);
        ArrayList<Integer>list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.add(5);
        list.add(6);
        // List = [1, 2, 3, 4, 5, 6]
        System.out.print("Enter a Target: ");
        int tar = sc.nextInt();
        System.out.println(pairSum1(list, tar));
    }
}
```

13. Size of ArrayList

```
import java.util.ArrayList;

public class SizeOfAL {
    public static void main(String[] args) {
        ArrayList<Integer>list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.add(5);
        System.out.println("Size of ArrayList is: " + list.size());
        // Traversing the ArrayList
        System.out.print("ArrayList is: ");
        for(int i = 0; i < list.size(); i++) {
            System.out.print(list.get(i) + " ");
        }
        System.out.println();

        // Reversing ArrayList
        System.out.print("Reverse of an ArrayList: ");
        for(int i = list.size()-1; i >= 0; i--) {
            System.out.print(list.get(i) + " ");
        }
        System.out.println();
    }
}
```

14. Sorting

```java
import java.util.*;

public class Sorting {
    public static void main(String[] args) {
        ArrayList<Integer>list = new ArrayList<>();
        list.add(2);
        list.add(5);
        list.add(9);
        list.add(3);
        list.add(6);
        System.out.print("Original List: ");
        System.out.println(list);
        System.out.print("Ascending Order: ");
        Collections.sort(list);
        System.out.println(list);
        System.out.print("Descending Order: ");
        Collections.sort(list, Collections.reverseOrder());
        System.out.println(list);
    }
}
```

15. Swapping

```java
import java.util.ArrayList;

public class Swapping {
    public static void swap (ArrayList<Integer>list, int idx1, int idx2) {
        int temp = list.get(idx1);
        list.set(idx1, list.get(idx2));
        list.set(idx2, temp);
    }
    public static void main(String[] args) {
        ArrayList<Integer>list = new ArrayList<>();
        list.add(2);
        list.add(5);
        list.add(9);
        list.add(3);
        list.add(6);
        int idx1 = 1, idx2 = 3;
        System.out.println(list);
        swap(list, idx1, idx2);
        System.out.println(list);
    }
}
```

16. Two Dimensional ArrayList

```java
import java.util.ArrayList;

public class TwoDarraylist {
    public static void main(String[] args) {
        ArrayList<ArrayList<Integer>>mainList = new ArrayList<>();
        ArrayList<Integer>list = new ArrayList<>();
        list.add(1);    list.add(2);
        mainList.add(list);
        ArrayList<Integer>list2 = new ArrayList<>();
        list2.add(3);   list2.add(4);
        mainList.add(list2);
        for(int i = 0; i<mainList.size(); i++) {
            ArrayList<Integer>currList = mainList.get(i);
            for(int j = 0; j < currList.size(); j++) {
                System.out.print(currList.get(j)+ " ");
            }
            System.out.println();
        }
        System.out.print("2D ArrayList is: ");
        System.out.println(mainList);
    }
}
```