ArrayLists

Creating an arraylist and basic Operations(add,get,set,remove,contains):

```java
import java.util.ArrayList;
import java.util.List;

public class creatingarraylist {
    public static void main(String args[]){
        //array list creation
        ArrayList <Integer> List = new ArrayList<>();
        //OPERATIONS like 1) set 2)get 3)contains 4)remove 5) add
        //Add Operation
        List.add(1); // or List.add(0,1);
        List.add(2);// or List.add(1,2);
        List.add(3);
        List.add(4);
        System.out.println( "add "+List);
        // set operation
        List.set(0,10);
        List.set(1,20);
        List.set(2,30);
        List.set(3,40);
        System.out.println("set"+ List);
        //get Operations
        System.out.print( "get "+List.get(0)+" ");
        System.out.print(List.get(1)+" ");
        System.out.print(List.get(2)+" ");
        System.out.print(List.get(3)+" ");
        // remove Operations
        List.remove(1);
        List.remove(2);
        System.out.println( "remove "+List);
        //contains operation
        System.out.println(List.contains(10));
        System.out.println(List.contains(11));



    }
```

```
}
Output:
add [1, 2, 3, 4]
set[10, 20, 30, 40]
get 10 20 30 40 remove [10, 30]
true
false
PS S:\A
```

Finding size of an arraylist:

```
import java.util.ArrayList;

public class sizeofarraylist {
    public static void main(String args[]){
        ArrayList<String> names= new ArrayList<>();
        names.add(0,"a");
        names.add(1,"b");
        names.add(2,"c");
        names.add(3,"d");
        System.out.println(names.size());
        for(int i=0;i<names.size();i++){
            System.out.println(names.get(i));
        }


    }

}
Output:
4
a
b
c
d
```

print reverse of an arraylist:

```
import java.util.*;
public class printreverseofanarraylist {
    public static void main(String args[]){
```

```
        ArrayList<Integer>l1=new ArrayList<>();
        l1.add(0,0);
        l1.add(1,1);
        l1.add(2,2);
        l1.add(3,3);
        l1.add(4,4);
        int N=l1.size();
        for(int i=N-1;i>=0;i--){
            System.out.println(l1.get(i));
        }

    }

}
Output:
4
3
2
1
0
```

Find maximum of an array:

```
import java.util.*;
public class findmaxinarraylist {
    public static void main(String args[]){
        ArrayList<Integer> list= new ArrayList<>();
        list.add(0,1);
        list.add(1,5);
        list.add(2,12);
        list.add(3,99);
        int max= Integer.MIN_VALUE;
        for(int i=0;i<list.size();i++){
            if(list.get(i)>max){
                max=list.get(i);
            }
        }
        System.out.println( " max value is: "+max);
    }


}
```

```
 max value is: 99
```

swaping two indexes:

```
import java.util.ArrayList;

public class swaptwoelements {
    public static void swap(ArrayList<Integer> l1,int index1,int index2){
        int temp = l1.get(index1);
        l1.set(index1,l1.get(index2));
        l1.set(index2,temp);

    }
    public static void main(String args[]){
        ArrayList<Integer> l1=new ArrayList<>();
        l1.add(0,1);
        l1.add(1,3);
        l1.add(2,4);
        l1.add(3,5);
        l1.add(4,6);
        //System.out.println(" before swap:"+"index1= "+l1.get(1)+" index3=
"+l1.get(3));
        int index1=1;
        int index2=3;
        // int temp= l1.get(index1);
        // l1.set(index1,l1.get(index2));
        // l1.set(index2,temp);
        //System.out.println(" after swap:"+"index1= "+l1.get(1)+" index3=
"+l1.get(3));
        System.out.println(l1);
        swap(l1, index1, index2);
        System.out.println(l1);



    }
}
Output:
[1, 3, 4, 5, 6]
[1, 5, 4, 3, 6]
```

Ascending and descending order in arraylists:

```
import java.util.Collections;
import java.util.ArrayList;
```

```java
public class sorting {
    public static void main(String args[]){
        ArrayList <Integer> l1= new ArrayList<>();
        l1.add(1);
        l1.add(7);
        l1.add(6);
        l1.add(5);
     // Print actual list
     System.out.println("Normal list: "+l1);
     // sorting ascending order
     Collections.sort(l1);
     System.out.println("ascending order: "+l1);
     // sorting descending order
     Collections.sort(l1,Collections.reverseOrder());
     System.out.println("descending order "+l1);
    }

}
Output:
Normal list: [1, 7, 6, 5]
ascending order: [1, 5, 6, 7]
descending order [7, 6, 5, 1]
```

multidimentional arraylists:

```java
 import java.util.*;
public class a2DarraylistsorMultidimentionalAl {
    public static void main(String args[]){
        ArrayList<ArrayList<Integer>> mainlist= new ArrayList<>();// creating
mainlist
        ArrayList<Integer> sublist1=new ArrayList<>();
        sublist1.add(1);
        sublist1.add(2);
        mainlist.add(sublist1);
        ArrayList<Integer> sublist2=new ArrayList<>();
        sublist2.add(3);
        sublist2.add(4);
        mainlist.add(sublist2);
        for(int i=0;i<mainlist.size();i++){
            ArrayList<Integer> currentlist=mainlist.get(i);
```

```
            for(int j=0;j<currentlist.size();j++){
                System.out.print(currentlist.get(j)+" ");
            }
            System.out.println();
        }
        System.out.println(mainlist);

    }

}
Output:
1 2
3 4
[[1, 2], [3, 4]]
```

Example 2:

```
import java.util.*;
public class multidimentionalarraylist {
    public static void main(String[] args) {
        ArrayList<ArrayList<Integer>> mainlist = new ArrayList<>();
        //sublists
        ArrayList<Integer> list1=new ArrayList<>();
        ArrayList<Integer> list2=new ArrayList<>();
        ArrayList<Integer> list3=new ArrayList<>();
        mainlist.add(list1);
        mainlist.add(list2);

        mainlist.add(list3);

        // adding elements int lists uding loop
        for(int i=1;i<=5;i++){
            list1.add(1*i);// 1 table
            list2.add(2*i);// 2 table
            list3.add(3*i);//3 table
       }
       // printing using 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();
```

```
        }
        //printing directly
        System.out.println(mainlist);
        list2.remove(0);
        list2.remove(1);
        System.out.println(mainlist);


    }

}
Output:
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
[[1, 2, 3, 4, 5], [2, 4, 6, 8, 10], [3, 6, 9, 12, 15]]
[[1, 2, 3, 4, 5], [4, 8, 10], [3, 6, 9, 12, 15]]
```

Maximum Water problem:

```
import java.util.*;
public class maximumwaterbruteforce {
    public static int maxwater(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.max(height.get(i),height.get(j));// height = max of 2
heights
                int weight=j-i;
                int currentwater=ht*weight;
                MaxWater=Math.max(MaxWater,currentwater);

            }
        }
        return MaxWater;
    }


    public static void main(String[] args) {

        ArrayList<Integer> height=new ArrayList<>();
        height.add(1);
        height.add(8);
```

```
        height.add(6);
        height.add(2);
        height.add(5);
        height.add(4);
        height.add(8);
        height.add(7);
        System.out.println(maxwater(height));

    }

}
Output: 49
```

Optimized code:

Timecomplexity=0(n):

```
import java.util.*;
public class maximumwateroptimized {
    public static int maxwater(ArrayList<Integer> height){
        //globally maxwater=0
        int MaxWat=0;
        int lp=0;
        int rp=height.size()-1;
        while(lp<rp){
        //calculate max water
        int ht=Math.max(height.get(lp),height.get(rp));
        int width= rp-lp;
        int currwater= ht*width;
        MaxWat=Math.max(currwater,MaxWat);

        // update pointer
        if(height.get(lp)<height.get(rp)){
            lp++;
        }
        else{
            rp--;
        }
    }
    return MaxWat;
}
    public static void main(String[] args) {
```

```
        ArrayList<Integer> height=new ArrayList<>();
        height.add(1);
        height.add(8);
        height.add(6);
        height.add(2);
        height.add(5);
        height.add(4);
        height.add(8);
        height.add(7);
        System.out.println(maxwater(height));

    }

}
Output:
49
```

Pairsum1 (brute force attack):

```
import java.util.*;
public class Pairsum1 {
 public static boolean Pairsum1(ArrayList<Integer> l1,int target){

    for(int i=0;i<l1.size();i++){
        for(int j=0;j<l1.size();j++){
            if(l1.get(i)+l1.get(j)==target){
                return true;

            }
        }
    }
    return false;
 }

    public static void main(String[] args) {
        ArrayList<Integer> l1= new ArrayList<>();
        l1.add(1);
        l1.add(2);
        l1.add(3);
        l1.add(4);
        l1.add(5);
        l1.add(6);
        int target =11;
        System.out.println(Pairsum1(l1, target));
```

```
    }

}
Output: true
```

Pairsum2 :

```
import java.util.*;
public class pairsum2 {
public static boolean Pairsum1(ArrayList<Integer> l1,int target){
    // case 1
    int breakingpoint= -1;
    for(int i=0;i<l1.size();i++){
        if(l1.get(i)>l1.get(i+1)){
            breakingpoint=i;
            break;
        }
    }
    int lp=breakingpoint+1;
    int rp=breakingpoint;
    // case 2 modular operations
    while(lp!=rp){
    if(l1.get(lp)+l1.get(rp)==target){
        return true;
    }
    else if(l1.get(lp)+l1.get(rp) <target){
        lp=(lp+1)%l1.size();
    }
    else{
        rp=(l1.size() +rp -1)%l1.size();
    }
    }
    return false;

}

    public static void main(String[] args) {
        ArrayList<Integer> l1= new ArrayList<>();
        l1.add(11);
        l1.add(15);
        l1.add(6);
        l1.add(8);
```

```
        l1.add(9);
        l1.add(10);
        int target =16;
        System.out.println(Pairsum1(l1, target));
    }

}
Output: true
```

Monotic arraylist:

```
import java.util.ArrayList;
import java.util.*;
public class monoticarraylist {
   public static boolean monotic(ArrayList<Integer> l1){
    int n=l1.size();
    boolean inc=true;
    boolean dec=true;
    for(int i=0;i<n-1;i++){
            if(l1.get(i)>l1.get(i+1)){
                inc = false;
            }
            else if(l1.get(i)<l1.get(i+1)){
                dec= false;
            }

    }
    return inc | dec;
   }

    public static void main(String[] args) {
        ArrayList<Integer> l1= new ArrayList<>();
        l1.add(1);
        l1.add(2);
        l1.add(2);
        l1.add(3);
      System.out.println(monotic(l1));
    }

}
Output: true
```

Lonely Numbers

```
import java.util.*;
public class lonelynumbers {
   public static ArrayList<Integer> lonelyNumbers( ArrayList<Integer>l1){
    Collections.sort(l1);
    ArrayList<Integer> l2= new ArrayList<>();
    int n=l1.size();

    for(int i=1;i<n-1;i++){
        if(l1.get(i-1)+1 < l1.get(i) && l1.get(i)+1 < l1.get(i+1)){
           l2.add(l1.get(i));
        }

         if(n==1){
            l2.add(l1.get(0));

         }
         if(n>1){
            if(l1.get(0)+1<l1.get(1)){
                l2.add(l1.get(0));
            }
            if(l1.get(n-2)+1<l1.get(n-1)){
                l2.add(l1.get(n-1));
            }
         }

    }
    for(int j=0;j<l2.size()-1;j++){
        System.out.println(l2.get(j));
    }
    return l2;
    public static void main(String[] args) {
        ArrayList<Integer>l1=new ArrayList<>();
        l1.add(10);
        l1.add(6);
        l1.add(5);
        l1.add(12);
        lonelyNumbers(l1);

    }
}
Output:
10
12
```

Most frequent Numbers

```
import java.util.*;
public class mostrepeatednumber {
public static int printans(ArrayList<Integer>l1 , int key){
    int count[]=new int[500];
    for(int i=0;i<l1.size()-1;i++){
        if(l1.get(i)==key){
            count[l1.get(i+1)-1]++;
        }
    }
    int max=Integer.MIN_VALUE;
    int ans=0;
    for(int i=0;i<500;i++){
        if(count[i]>max){
            max=count[i];
            ans=i+1;
        }
    }
    return ans;


}

    public static void main(String args[])
    {
        ArrayList<Integer>l1=new ArrayList<>();
        l1.add(1);
        l1.add(100);
        l1.add(200);
        l1.add(1);
        l1.add(100);
        l1.add(200);
        int key=1;
        System.out.println(printans(l1, key));

    }

}
Output:100
```

Beautiful ArrayList:

```
import java.util.*;
public class beautifularraylist {
    public ArrayList<Integer> beautiful(int n){
        ArrayList<Integer> result=new ArrayList<>();
        Divideandconquer(1, 1, result, n);


        return result;}

        public static void print( ArrayList<Integer> result){
            for(int i=0;i<result.size();i++){
                System.out.print(result.get(i));
        }

    }
    public static void Divideandconquer( int start, int
increment,ArrayList<Integer> result,int n){
        if(start + increment>n){
            result.add(start);
            return;
        }
        Divideandconquer(start, 2*increment, result, n);
        Divideandconquer(start+increment, 2*increment, result, n);
    }
    public static void main(String args[]){
        ArrayList<Integer> result=new ArrayList<>();
        int start, increment,n;
        Divideandconquer(1, 1, result, 5);

        print(result);


    }
}
Output:53241
```