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

public class main {
   //Part A
   public static void main(String[] args){
       main m=new main();
       System.out.println("-----Lab 3 Part A Tester-----");
       int min=-100;
       int max=100;
       System.out.println("Please enter a positive interger:");
       Scanner scan=new Scanner(System.in);
       int n=scan.nextInt();
       int[] array=new int[n];
       for(int i=0;i<n;i++){
           array[i]=(int)(Math.random()*(max-min+1)+min);
       }
       System.out.println("Generated Array: "+Arrays.toString(array));
       System.out.println("Please enter a number between 1 and "+n+":");
       int k=scan.nextInt();
       System.out.println(m.Quick_select(array,k)+"\n");
       System.out.println("-----Lab 3 Part B Tester-----");
       int[] b={4,2,0,10,1,6};
       System.out.println(Arrays.toString(m.Quick_selectMax(b,3)));


   }

   /*
   Quick_select function:
   Take an array and a number as its arguments
   return the smallest k-th number in the list.
   */
   public int Quick_select(int[] a, int k){
       int piDex=Partition(a);
       //System.out.println(piDex);
       //System.out.println(Arrays.toString(a));
       //System.out.println(k);
       if (k-1==piDex){
           //System.out.println(a[piDex]);
           return a[piDex];
       }
       else if(k-1<piDex){

           return Quick_select(Arrays.copyOfRange(a,0,piDex),k);
       }

       else if(k-1>piDex){
```

```
            return
Quick_select(Arrays.copyOfRange(a,piDex+1,a.length),k-piDex-1);

        }
        return -1;
    }

    /*
    Partition function:
    Separate an array into two part that any of the left side will be smaller
than the right side
    */
    private int Partition(int[] a){
        int len=a.length-1;
        int pivot = MedianofThree(a[0],a[(int)(a.length/2)],a[len]);
        //System.out.println(pivot);
        //System.out.println((int)(a.length/2));
        int left=0;
        int right=len;
        int tempNum;
        //System.out.println(Arrays.toString(a));
        while(left<right){

            while(a[left]<pivot){
                left++;
            }

            while(a[right]>pivot){
                right--;
            }
            //System.out.println(left+" "+right);
            if(left>=right){
                break;
            }

            tempNum=a[left];
            a[left]=a[right];
            a[right]=tempNum;

        }
        //System.out.println(Arrays.toString(a));
        return left;

    }

    /*
    Medianofthree function:
    Find the median from tree number
    */
```

```
private int MedianofThree(int a, int b, int c){
    if(a>b & a<c){
        return a;
    }
    else if(b>a & b<c){
        return b;
    }
    else{
        return c;
    }
}


/*
Part B
Quick_selectMax function:
Take an array and a number as its arguments
return the largest k numbers in the list.
*/
public int[] Quick_selectMax(int[] a, int k){
    int pivot= Quick_select(a,a.length-k);
    return Partition(a,pivot);
}

/*
Modified Partition function
Take an array and a pivot point as its arguments
return the right array from the pivot
*/
private int[] Partition(int[] a, int n){
    int len=a.length-1;
    int pivot = n;
    //System.out.println(pivot);
    //System.out.println((int)(a.length/2));
    int left=0;
    int right=len;
    int tempNum;
    //System.out.println(Arrays.toString(a));
    while(left<right){

        while(a[left]<pivot){
            left++;
        }

        while(a[right]>pivot){
            right--;
        }
        //System.out.println(left+" "+right);
        if(left>=right){
```

```
                break;
            }

            tempNum=a[left];
            a[left]=a[right];
            a[right]=tempNum;

        }
        //System.out.println(Arrays.toString(a));
        return Arrays.copyOfRange(a,right+1,a.length);
    }

}
```