```java
import java.util.Random;
import java.util.Scanner;

public class MergeArray {
    public static Random rand = new Random();
    public static int i = 0;
    public static int j = 0;
    public static int k = 0;

    // returns true if a number exist in the given array
    public static boolean exist(int[] arr, int x){
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == x)
                return true;
        }
        return false;
    }

    // returns a new random array with unique random numbers between min and max values
    public static int[] getRandomArray(int arrLength, int min, int max){
        int[] arr = new int[arrLength];
        int temp;
        for (int i = 0; i < arr.length; i++)
        {
            do {
                temp = rand.nextInt(min, max);
            }while (exist(arr,temp));
            arr[i] = temp;
        }
        return arr;
    }

    // prints an array
    public static void printArr(int[] arr, int arrLength){
        for (int i = 0; i < arrLength; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }

    // Sorts an array in bubble sort fashion
    public static void bubbleSort(int[] arr) {
        int i = arr.length - 1;
        boolean sorted = false;

        while (!sorted && i > 0) {
            sorted = true;
            for (int j = 0; j < i; j++) {
```

```
49              if (arr[j] > arr[j + 1]) {
50                  swap(arr, j, j + 1);
51                  sorted = false;
52              }
53          }
54          i--;
55      }
56  }
57
58  // swaps 2 numbers in the array
59  public static void swap(int[] arr, int x, int y) {
60      int temp = arr[x];
61      arr[x] = arr[y];
62      arr[y] = temp;
63  }
64
65  // Sorts an array in selection sort fashion
66  public static void selectionSort(int[] arr){
67      int p;
68      for (int i = arr.length -1; i > 0 ; i--) {
69          p = maxPlaceInArr(arr, i);
70          swap(arr, i, p);
71      }
72  }
73  // פעולה שמקבלת מערך ומחזירה את מיקומו של האיבר הכי גדול במערך עד מקום זה
74  public static int maxPlaceInArr(int[] arr, int place){
75      int max = 0;
76      for (int i = 1; i <= place; i++) {
77          if (arr[i] > arr[max])
78              max = i;
79      }
80      return max;
81  }
82
83
84  // merges two sorted array into a bigger array and returns it and removes
    any duplicate numbers
85  public static int[] mergeArrs(int[] a, int[] b){
86      int[] c = new int[a.length+b.length];
87      i=0;
88      j=0;
89      k=0;
90      while (i < a.length && j < b.length ){
91          if(!exist(c, a[i])){
92              if(!exist(c, b[j])){
93                  if(a[i] < b[j])
94                  {
95                      c[k] = a[i];
96                      i++;
```

```
 97                 } else {
 98                     c[k] = b[j];
 99                     j++;
100                 }
101                 k++;
102             } else j++;
103         } else i++;
104 
105     }
106     while (i<a.length){
107         if(!exist(c, a[i])){
108             c[k] = a[i];
109             k++;
110         }
111         i++;
112     }
113     while (j<b.length){
114         if(!exist(c, b[j])){
115             c[k] = b[j];
116             k++;
117         }
118         j++;
119     }
120     return c;
121 }
122 
123 // merges two sorted array into a bigger array and returns it
124 public static int[] mergeArr(int[] a, int[] b){
125     int[] c = new int[a.length+b.length];
126     i=0;
127     j=0;
128     k=0;
129     while (i < a.length && j < b.length ){
130         if(a[i] < b[j])
131         {
132             c[k] = a[i];
133             i++;
134         } else {
135             c[k] = b[j];
136             j++;
137         }
138         k++;
139     }
140     while (i<a.length){
141         c[k] = a[i];
142         i++;
143         k++;
144     }
145     while (j<b.length){
```

```
146             c[k] = b[j];
147             j++;
148             k++;
149         }
150         return c;
151     }
152
153     // returns a sorted array that consists of numbers who appear in both
        arrays
154     // Function to find the intersection of two sorted arrays
155     public static int[] intersectArr(int[] a, int[] b) {
156         i = 0;
157         j = 0;
158         k = 0;
159         int[] c = new int[a.length + b.length];
160         while (i < a.length && j < b.length) {
161             if (a[i] < b[j]) {
162                 i++;
163             } else if (a[i] > b[j]) {
164                 j++;
165             } else {
166                 c[k] = a[i];
167                 i++;
168                 j++;
169                 k++;
170             }
171         }
172         return c;
173     }
174
175     public static void main(String[] args) {
176         Scanner input = new Scanner(System.in);
177
178         System.out.print("enter the size of the first array --> ");
179         int n = input.nextInt();
180         System.out.print("enter the size of the 2nd array --> ");
181         int m = input.nextInt();
182
183         int[] a = getRandomArray(n, 1, 20);
184         int[] b = getRandomArray(m, 1, 100);
185         printArr(a, a.length);
186         printArr(b, b.length);
187
188         bubbleSort(a);
189         printArr(a, a.length);
190         selectionSort(b);
191         printArr(b, b.length);
192
193         int[] c = mergeArr(a, b);
```

```
194         printArr(c, c.length);
195
196         int[] d = mergeArrs(a,b);
197         printArr(d, k);
198
199         int[] e = intersectArr(a,b);
200         printArr(e, k);
201     }
202 }
203 /*
204 enter the size of the first array --> 7
205 enter the size of the 2nd array --> 9
206
207 1 13 3 16 19 12 6
208 54 11 6 99 20 43 32 59 9
209
210 1 3 6 12 13 16 19
211 6 9 11 20 32 43 54 59 99
212
213 1 3 6 6 9 11 12 13 16 19 20 32 43 54 59 99
214 1 3 6 9 11 12 13 16 19 20 32 43 54 59 99
215 6
216 * */
217
218
```