```java
/*
 * Cynthia C.
 * 15th April, 2020
 * sorting numbers
 */
package sorting;

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

public class SortingMain {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int userChoice;
        int[] list = new int[100];
        int[] list2 = new int[100];

        //will loop until user wishes to exit
        do {
            System.out.println("1) selection sort\n2) insertion sort\n3) bubble sort\n4) exit");
            userChoice = input.nextInt();

            genArray(list);

            //sets the second list for the before sorting
            for(int t = 0; t < list.length; t++) {
                list2[t] = list[t];
            }

            //will choose an option based on userChoice
            if(userChoice == 1) {
                list = selection(list);
            }else if(userChoice == 2) {
                list = insertion(list);
            }else if(userChoice == 3) {
                list = bubble(list);
            }else if(userChoice == 4) {
                System.out.println("Bye!");
            }else {
                System.out.println("That is not an acceptable choice\nPlease choose again");
            }

            //prints out the list before and after
            if(userChoice > 0 && userChoice < 4) {
                System.out.format("%s %s%n", "BEFORE SORT", "AFTER SORT");
                for(int i = 0; i < list.length; i++) {
                    System.out.format("%7s %12s%n", list2[i], list[i]);
                }
            }
        }while(userChoice != 4);

    }

    /**
     * will generate 100 random numbers between 1-1000
```

```
57      * will return the array
58      * @param list
59      * @return
60      */
61     public static int[] genArray(int[] list) {
62         Random r = new Random();
63 
64         //loops for the length of the array(100 times)
65         for(int i = 0; i < list.length; i++) {
66             list[i] = r.nextInt(1000 - 1) + 1;
67         }
68 
69         return list;
70     }
71 
72     /**
73      * will compare the first number in an array with every number
74      * in the array until it finds one smaller than itself
75      * will then swap both those numbers
76      * returns the list
77      * @param list
78      * @return
79      */
80     public static int[] selection(int[] list) {
81         int minV, minI, swap = 0;
82 
83         //loops for the length of the array
84         for(int i = 0; i < list.length; i++) {
85             minV = list[i];
86             minI = i;
87             //loops for the length of the array to compare every number
88             for(int j = i; j < list.length; j++) {
89                 //will change the minimum value if it finds one smaller than the first value
   in the array
90                 if(list[j] < minV) {
91                     minV = list[j];
92                     minI = j;
93                 }
94             }
95             //if the value is smaller than the first in the array it will swap the two numbers
96             if(minV < list[i]) {
97                 swap = list[i];
98                 list[i] = list[minI];
99                 list[minI] = swap;
100            }
101        }
102 
103        return list;
104    }
105 
106    /**
107     * will compare a number with all the ones to it's left
108     * if the one to the left is smaller it will swap them
109     * returns list
110     * @param list
111     * @return
112     */
```

```java
    public static int[] insertion(int[] list) {
        int mc, swap, sk;

        //loops for the length of the array
        for(int i = 1; i < list.length; i++) {
            mc = list[i];
            sk = i - 1;
            //will loop until there are no more numbers to compare with or it has reached a number smaller than itself
            while(sk >= 0 && mc < list[sk]) {
                swap = list[sk];
                list[sk] = list[sk + 1];
                list[sk + 1] = swap;
                sk--;
            }
        }

        return list;
    }

    /**
     * compares pairs of numbers next to each other
     * the smaller # will move to the left
     * with each loop through the array the one that ends up furthest right will be the largest
     * so the array length will shorten one bc it doesn't need to check it anymore
     * @param list
     * @return
     */
    public static int[] bubble(int[] list) {
        int swap = 0;

        //loops the length of the array - 1
        for(int i = 0; i < list.length - 1; i++) {
            //loops the length of the array subtract the number of spots the computer knows are correctly placed
            for(int j = 0; j < list.length - 1 - i; j++) {
                //will swap if one on left is higher
                if(list[j] > list[j + 1]) {
                    swap = list[j];
                    list[j] = list[j + 1];
                    list[j + 1] = swap;
                }
            }
        }

        return list;
    }

}

```