```java
package TestingAndComplexity;
import java.util.Random;
public class IntSortSearchTester {
    //non-static random object
    Random rand = new Random();
    int randomInt = rand.nextInt(100 + 100) -100;

    //method which just calls all the tester methods, using a statement like: IntSortSearch.linearSearchTester();
    public static void main(String [] args) {
        //Test Case 1, 2, and 3 for linearSearch method
        System.out.println("Testing for linearSearch of the IntSortSearch class\n");
        for (int count = 0; count < 3; count++) {
            IntSortSearchTester.linearSearchTester();
            System.out.println("\n*****************************************\n");
        }

        //Test Case 1, 2, and 3 for binarySearch method
        System.out.println("Testing for binarySearch of the IntSortSearch class\n");
        for (int count = 0; count < 3; count++) {
            IntSortSearchTester.binarySearchTester();
            System.out.println("\n*****************************************\n");
        }

        //Test Case 1, 2, and 3 for selectionSort method
        System.out.println("Testing for selectionSort of the IntSortSearch class\n");
        for (int count = 0; count < 3; count++) {
            IntSortSearchTester.sortTester();
            System.out.println("\n*****************************************\n");
        }
    }

    //This hard-codes the test data for linearSearch method
    public static void linearSearchTester()  {
        //prompts
        String prompt1 = "Array containing one element only, and the value being searched for is the element in the array";
        String prompt2 = "Array containing one element only, and the value being searched for is NOT the element in the array";
        String prompt3 = "Array containing two unequal elements, and the value being searched for is one of the elements of the array";
        String prompt4 = "Array containing two unequal elements, and the value being searched for is NOT ANY of the elements of the array";
        String prompt5 = "Array containing 3 elements, and the value being searched for is the very first element of the array";
        String prompt6 = "Array containing 3 elements, and the value being searched for is the very last element of the array";
        String prompt7 = "Array containing 3 elements, and the value being searched for is the middle element";
```

```
42          String prompt8 = "Array containing 3 elements, and the value being searched
    for is NOT ANY of the elements in the array";
43          String [] prompts = new String [] {prompt1, prompt2, prompt3, prompt4,
    prompt5, prompt6, prompt7, prompt8};
44
45          //declare all variables
46          int [] oneElemA, oneElemB, twoElemA, twoElemB, threeElemA, threeElemB,
    threeElemC, threeElemD;
47          int oneElem1, oneElem2, twoElem1, twoElem2, threeElem1, threeElem2,
    threeElem3, threeElem4;
48
49          //all random integers are in the range of (-99 , 99)
50          Random rand = new Random();
51
52          //Array containing one element only, and the value being
53          //searched for is the element in the array
54          oneElemA = new int[] {rand.nextInt(100 + 100) -100};
55          oneElem1 = oneElemA[0];
56
57          //Array containing one element only, and the value being
58          //searched for is NOT the element in the array
59          oneElemB = new int[] {rand.nextInt(100 + 100) -100};
60          oneElem2 = rand.nextInt(100 + 100) -100;
61          //make sure that the element is not in the array
62          while (oneElem2 == oneElemB[0])
63              oneElem2 = rand.nextInt(100 + 100) -100;
64
65          //Array containing two unequal elements, and the value being
66          //searched for is one of the elements of the array
67          twoElemA = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 + 100)
    -100};
68          //make sure that the elements in the array are not equal to each other
69          while (twoElemA[0] == twoElemA[1])
70              twoElemA[1] = rand.nextInt(100 + 100) -100;
71          twoElem1 = twoElemA[0];
72
73          //Array containing two unequal elements, and the value being
74          //searched for is NOT ANY of the elements of the array
75          twoElemB = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 + 100)
    -100};
76          //make sure that the elements in the array are not equal to each other
77          while (twoElemB[0] == twoElemB[1])
78              twoElemB[1] = rand.nextInt(100 + 100) -100;
79          twoElem2 = rand.nextInt(100 + 100) -100;
80          while ((twoElem2 == twoElemB[0]) || (twoElem2 == twoElemB[1]))
81              twoElem2 = rand.nextInt(100 + 100) -100;
82
83          //Array containing 3 elements, and the value being
84          //searched for is the very first element of the array
```

```
85          threeElemA = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 +
    100) -100, rand.nextInt(100 + 100) -100};
86          threeElem1 = threeElemA[0];
87
88          //Array containing 3 elements, and the value being
89          //searched for is the very last element of the array
90          threeElemB = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 +
    100) -100, rand.nextInt(100 + 100) -100};
91          threeElem2 = threeElemB[2];
92
93          //Array containing 3 elements, and the value being
94          //searched for is the middle element
95          threeElemC = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 +
    100) -100, rand.nextInt(100 + 100) -100};
96          threeElem3 = threeElemC[1];
97
98          //Array containing 3 elements, and the value being
99          //searched for is NOT ANY of the elements in the array
100         threeElemD = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 +
    100) -100, rand.nextInt(100 + 100) -100};
101         threeElem4 = rand.nextInt(100 + 100) -100;
102         //make sure that the element is not in the array
103         while ((threeElem4 == threeElemD[0]) || (threeElem4 == threeElemD[1]) ||
    (threeElem4 == threeElemD[2]))
104             threeElem4 = rand.nextInt(100 + 100) -100;
105
106         int [] arrayOfElements = {oneElem1, oneElem2, twoElem1, twoElem2,
    threeElem1, threeElem2, threeElem3, threeElem4};
107         int [] [] allArrays = {oneElemA, oneElemB, twoElemA, twoElemB, threeElemA,
    threeElemB, threeElemC, threeElemD};
108
109         int element;
110         int [] array;
111         int position;
112         for (int count = 0; count < 8; count++) {
113             element = arrayOfElements[count];
114             array = allArrays[count];
115
116             position = IntSortSearch.binarySearch(array, element);
117             String output;
118             if (position >= 0){
119                 output = ("In searching for " + element + " in the " +
    prompts[count]);
120                 output += (", aka " + toString(allArrays[count]) + ", " + element +
    " was found at position " + position + ".");
121                 System.out.println(output);
122             }
123             else {
124                 output = ("In searching for " + element + " in the " +
    prompts[count]);
```

```
125                 output += (", aka " + toString(allArrays[count]) + ", " + element +
    " was not found.");
126                 System.out.println(output);
127             }
128         }
129     }
130 
131     //This hard-codes the test data for binarySearch method
132     public static void binarySearchTester(){
133         //prompts
134         String prompt1 = "Array containing one element only, and the value being
    searched for is the element in the array";
135         String prompt2 = "Array containing one element only, and the value being
    searched for is NOT the element in the array";
136         String prompt3 = "Array containing two unequal elements, and the value being
    searched for is one of the elements of the array";
137         String prompt4 = "Array containing two unequal elements, and the value being
    searched for is NOT ANY of the elements of the array";
138         String prompt5 = "Array containing 3 elements, and the value being searched
    for is the very first element of the array";
139         String prompt6 = "Array containing 3 elements, and the value being searched
    for is the very last element of the array";
140         String prompt7 = "Array containing 3 elements, and the value being searched
    for is the middle element";
141         String prompt8 = "Array containing 3 elements, and the value being searched
    for is NOT ANY of the elements in the array";
142         String [] prompts = new String [] {prompt1, prompt2, prompt3, prompt4,
    prompt5, prompt6, prompt7, prompt8};
143 
144         //declare all variables
145         int [] oneElemA, oneElemB, twoElemA, twoElemB, threeElemA, threeElemB,
    threeElemC, threeElemD;
146         int oneElem1, oneElem2, twoElem1, twoElem2, threeElem1, threeElem2,
    threeElem3, threeElem4;
147 
148         //all random integers are in the range of (-99 , 99)
149         Random rand = new Random();
150 
151         //Array containing one element only, and the value being
152         //searched for is the element in the array
153         oneElemA = new int[] {rand.nextInt(100 + 100) -100};
154         oneElem1 = oneElemA[0];
155 
156         //Array containing one element only, and the value being
157         //searched for is NOT the element in the array
158         oneElemB = new int[] {rand.nextInt(100 + 100) -100};
159         oneElem2 = rand.nextInt(100 + 100) -100;
160         //make sure that the element is not in the array
161         while (oneElem2 == oneElemB[0])
162             oneElem2 = rand.nextInt(100 + 100) -100;
```

```
163
164         //Array containing two unequal elements, and the value being
165         //searched for is one of the elements of the array
166         twoElemA = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 + 100)
    -100};
167         //make sure that the elements in the array are not equal to each other
168         while (twoElemA[0] == twoElemA[1])
169             twoElemA[1] = rand.nextInt(100 + 100) -100;
170         twoElem1 = twoElemA[0];
171
172         //Array containing two unequal elements, and the value being
173         //searched for is NOT ANY of the elements of the array
174         twoElemB = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 + 100)
    -100};
175         //make sure that the elements in the array are not equal to each other
176         while (twoElemB[0] == twoElemB[1])
177             twoElemB[1] = rand.nextInt(100 + 100) -100;
178         twoElem2 = rand.nextInt(100 + 100) -100;
179         while ((twoElem2 == twoElemB[0]) || (twoElem2 == twoElemB[1]))
180             twoElem2 = rand.nextInt(100 + 100) -100;
181
182         //Array containing 3 elements, and the value being
183         //searched for is the very first element of the array
184         threeElemA = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 +
    100) -100, rand.nextInt(100 + 100) -100};
185         threeElem1 = threeElemA[0];
186
187         //Array containing 3 elements, and the value being
188         //searched for is the very last element of the array
189         threeElemB = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 +
    100) -100, rand.nextInt(100 + 100) -100};
190         threeElem2 = threeElemB[2];
191
192         //Array containing 3 elements, and the value being
193         //searched for is the middle element
194         threeElemC = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 +
    100) -100, rand.nextInt(100 + 100) -100};
195         threeElem3 = threeElemC[1];
196
197         //Array containing 3 elements, and the value being
198         //searched for is NOT ANY of the elements in the array
199         threeElemD = new int[] {rand.nextInt(100 + 100) -100, rand.nextInt(100 +
    100) -100, rand.nextInt(100 + 100) -100};
200         threeElem4 = rand.nextInt(100 + 100) -100;
201         //make sure that the element is not in the array
202         while ((threeElem4 == threeElemD[0]) || (threeElem4 == threeElemD[1]) ||
    (threeElem4 == threeElemD[2]))
203             threeElem4 = rand.nextInt(100 + 100) -100;
204
```

```
205         int [] arrayOfElements = {oneElem1, oneElem2, twoElem1, twoElem2,
    threeElem1, threeElem2, threeElem3, threeElem4};
206         int [] [] allArrays = {oneElemA, oneElemB, twoElemA, twoElemB, threeElemA,
    threeElemB, threeElemC, threeElemD};
207 
208         //sort all arrays
209         for (int count = 0; count < allArrays.length; count++) {
210             IntSortSearch.selectionSort(allArrays[count]);
211         }
212 
213         int element;
214         int [] array;
215         int position;
216         for (int count = 0; count < 8; count++) {
217             element = arrayOfElements[count];
218             array = allArrays[count];
219 
220             position = IntSortSearch.binarySearch(array, element);
221             String output;
222             if (position >= 0){
223                 output = ("In searching for " + element + " in the " +
    prompts[count]);
224                 output += (", aka " + toString(allArrays[count]) + ", " + element +
    " was found at position " + position + ".");
225                 System.out.println(output);
226             }
227             else {
228                 output = ("In searching for " + element + " in the " +
    prompts[count]);
229                 output += (", aka " + toString(allArrays[count]) + ", " + element +
    " was not found.");
230                 System.out.println(output);
231             }
232         }
233     }
234 
235     //This hard-codes the test data for selectionSort method
236     public static void sortTester() {
237         //prompts
238         String prompt1 = "Array containing one element only";
239         String prompt2 = "Array containing two unequal elements";
240         String prompt3 = "Array containing 3 elements, already sorted";
241         String prompt4 = "Array containing 3 elements, already sorted in the reverse
    order";
242         String prompt5 = "Array containing 3 elements, all of them negative";
243         String prompt6 = "Array containing 3 elements, one negative, one zero, one
    positive";
244         String prompt7 = "Array containing 3 elements, all of them the same value
    (like: 5,5,5)";
```

```
245         String [] prompts = new String [] {prompt1, prompt2, prompt3, prompt4,
    prompt5, prompt6, prompt7};
246         //declare all variables
247         int [] oneElemA,twoElemA, threeElemA, threeElemB, threeElemC, threeElemD,
    threeElemE;
248
249         Random rand = new Random();
250         //Array containing one element only
251         oneElemA = new int[] {rand.nextInt(100 + 100) - 100};
252
253         //Array containing two unequal elements
254         twoElemA = new int[] {rand.nextInt(100 + 100) - 100, rand.nextInt(100 + 100)
    - 100};
255         //make sure that the elements in the array are not equal to each other
256         while (twoElemA[0] == twoElemA[1])
257             twoElemA[1] = rand.nextInt(100 + 100) - 100;
258
259         //Array containing 3 elements, already sorted
260         threeElemA = new int[] {rand.nextInt(100 + 100) - 100, rand.nextInt(100 +
    100) - 100, rand.nextInt(100 + 100) - 100};
261         bubbleSort(threeElemA);
262
263         //Array containing 3 elements, already sorted in the reverse order
264         threeElemB = new int[] {rand.nextInt(100 + 100) - 100, rand.nextInt(100 +
    100) - 100, rand.nextInt(100 + 100) - 100};
265         bubbleSort(threeElemB);
266         //reverse order
267         int temp;
268         temp = threeElemB[0];
269         threeElemB[0] = threeElemB[2];
270         threeElemB[2] = temp;
271
272         //Array containing 3 elements, all of them negative
273         threeElemC = new int[] {rand.nextInt(100) - 100, rand.nextInt(100) - 100,
    rand.nextInt(100) - 100};
274
275         //Array containing 3 elements, one negative, one zero, one positive
276         threeElemD = new int[] {rand.nextInt(100) - 100, 0, rand.nextInt(100)};
277
278         //Array containing 3 elements, all of them the same value (like: 5,5,5)
279         int perm = rand.nextInt(100 + 100) - 100;
280         threeElemE = new int[] {perm, perm, perm};
281
282         int [] [] allArrays = {oneElemA,twoElemA, threeElemA, threeElemB,
    threeElemC, threeElemD, threeElemE};
283         int [] array, expectedArray, ogArray;
284         String output;
285         for (int count = 0; count < 7; count++) {
286             array = allArrays[count];
287
```

```
288             //preserve original array
289             ogArray = array.clone();
290 
291             //expected output
292             expectedArray = array.clone();
293             bubbleSort(expectedArray);
294 
295             //test the selectionSort method
296             IntSortSearch.selectionSort(array);
297             //print statements
298             output = ("In sorting the " + prompts[count] + ", aka " +
    toString(ogArray));
299             output += (", the expected array is " + toString(expectedArray));
300             output += (". The actual sorted array using the selectionSort method is
    " + toString(array) + ".");
301             System.out.println(output);
302         }
303     }
304 
305     public static void bubbleSort(int[] a) {
306         boolean sorted = false;
307         int temp;
308         while(!sorted) {
309             sorted = true;
310             for (int i = 0; i < a.length - 1; i++) {
311                 if (a[i] > a[i+1]) {
312                     temp = a[i];
313                     a[i] = a[i+1];
314                     a[i+1] = temp;
315                     sorted = false;
316                 }
317             }
318         }
319     }
320 
321     public static String toString(int [] a) {
322         String answer = "[";
323         for (int count = 0; count < a.length; count++) {
324             answer += (a[count] + ", ");
325         }
326         answer = answer.substring(0, answer.length() - 2);
327         answer += "]";
328         return answer;
329     }
330 }
```