

```

1  package esof322hw2;
2
3  import java.util.Arrays;
4  import java.util.Scanner;
5
6  /**
7   * ESOF322 HW 2 Client to choose math software
8   * @author Karl Molina, Dana Parker
9   */
10 public class Client {
11     /**
12      * Main method that lets you select a math software to sort numbers
13      * with a default sort strategy
14      * then keep that same math software but change the sort strategy
15      * @param args
16      */
17     public static void main(String[] args) {
18         //numbers to sort
19         int[] numbers = {9, 3, 6, 8, 7};
20         Scanner in = new Scanner(System.in);
21
22         //mathSoftware that holds the sorting strategies
23         MathSoftware mathSoftware = null;
24         System.out.println("You must sort these numbers: " + Arrays.toString(numbers));
25         System.out.println("Which math software would you like to use to sort them?");
26         System.out.println("Mathematica (1), MTool (2), or MyMath (3)?");
27
28         //input chooses which software to use
29         switch (in.nextInt()) {
30             case 1:
31                 mathSoftware = new Mathematica();
32                 break;
33             case 2:
34                 mathSoftware = new MTool();
35                 break;
36             case 3:
37                 mathSoftware = new MyMath();
38                 break;
39             default:
40                 System.out.println("You have entered an invalid number. Program
41                 terminated.");
42                 in.close();
43                 return;
44         }
45
46         System.out.println("Executing " + mathSoftware + "'s default sorting
47         algorithm...");
48         mathSoftware.mathSort(numbers);
49         System.out.println("Your result is: " + Arrays.toString(numbers));
50
51         //new numbers to sort
52         numbers = new int[] {5, 3, 1, 7, 2};
53         System.out.println("\nYou must now sort these numbers: " +
54         Arrays.toString(numbers));
55         System.out.println("Which new sorting algorithm would you like to use?");
56         System.out.println("InsertionSort (1), MergeSort (2), or BubbleSort (3)?");
57
58         //input chooses which sort strategy to switch to
59         switch (in.nextInt()) {
60             case 1:
61                 mathSoftware.setSortStrategy(new InsertionSort());
62                 break;
63             case 2:
64                 mathSoftware.setSortStrategy(new MergeSort());
65                 break;
66             case 3:
67                 mathSoftware.setSortStrategy(new BubbleSort());
68                 break;
69             default:

```

```

67         System.out.println("You have entered an invalid number. Program
68         terminated.");
69         in.close();
70         return;
71     }
72     in.close();
73     System.out.println("Executing new sorting algorithm...");
74     mathSoftware.mathSort(numbers);
75     System.out.println("Your result is: " + Arrays.toString(numbers));
76 }
77 }
78
79 /**
80  * Superclass from which Mathematica, MTool, and MyMath extends.
81  * Defines setSortStrategy(), and mathSort().
82  * Contains an instance of ISortStrategy.
83  */
84 abstract class MathSoftware {
85
86     protected ISortStrategy sortStrategy;
87
88     protected MathSoftware(ISortStrategy sortStrategy) {
89         System.out.println("Creating a Math Software: ");
90         this.sortStrategy = sortStrategy;
91     }
92
93     public void setSortStrategy(ISortStrategy sortStrategy) {
94         System.out.println("Setting your math software's (" + this + ") sorting
95         algorithm to " + sortStrategy);
96         this.sortStrategy = sortStrategy;
97     }
98
99     public void mathSort(int[] input) {
100         System.out.println("Sorting the numbers using " + sortStrategy);
101         sortStrategy.sort(input);
102     }
103 }
104
105 /**
106  * A math software that has the default sort strategy of insertion sort.
107  */
108 class Mathematica extends MathSoftware {
109
110     public Mathematica() {
111         super(new InsertionSort());
112         System.out.println("\tInitializing Mathematica");
113     }
114
115     @Override
116     public String toString() {
117         return "Mathematica";
118     }
119 }
120
121 /**
122  * A math software that has the default sort strategy of merge sort.
123  */
124 class MTool extends MathSoftware {
125
126     public MTool() {
127         super(new MergeSort());
128         System.out.println("\tInitializing MTool");
129     }
130
131     @Override
132     public String toString() {
133         return "MTool";
134     }
135 }

```

```

134     }
135
136     /**
137      * A math software that has the default sort strategy of bubble sort.
138      */
139     class MyMath extends MathSoftware {
140
141         public MyMath() {
142             super(new BubbleSort());
143             System.out.println("\tInitializing MyMath");
144         }
145
146         @Override
147         public String toString() {
148             return "MyMath";
149         }
150     }
151
152     /**
153      * Interface that defines the method sort().
154      */
155     interface ISortStrategy {
156
157         public void sort(int[] input);
158     }
159
160     /**
161      * The class that represents insertion sort.
162      * Implements ISortStrategy with insertion sort.
163      * Implementation from https://www.geeksforgeeks.org/insertion-sort/
164      */
165     class InsertionSort implements ISortStrategy {
166
167         /**
168          * Implementation of Insertion Sort taken from
169          * https://www.geeksforgeeks.org/insertion-sort/
170          */
171         @Override
172         public void sort(int[] arr) {
173             System.out.println("Insertion Sort executed.");
174             int n = arr.length;
175             for (int i=1; i<n; ++i)
176             {
177                 int key = arr[i];
178                 int j = i-1;
179
180                 /* Move elements of arr[0..i-1], that are
181                  greater than key, to one position ahead
182                  of their current position */
183                 while (j>=0 && arr[j] > key)
184                 {
185                     arr[j+1] = arr[j];
186                     j = j-1;
187                 }
188                 arr[j+1] = key;
189             }
190         }
191
192         @Override
193         public String toString() {
194             return "Insertion Sort";
195         }
196     }
197
198     /**
199      * The class that represents merge sort.
200      * Implements ISortStrategy with merge sort.
201      * Implementation from https://www.geeksforgeeks.org/merge-sort/
202      */

```

```

203 class MergeSort implements ISortStrategy {
204
205     /**
206      * Sort function that uses the help merge sort functions
207      */
208     @Override
209     public void sort(int[] input) {
210         System.out.println("Merge Sort executed.");
211         sort(input, 0, input.length-1);
212     }
213
214     /**
215      * Sort helper function
216      * Implementation of merge sort
217      * taken from https://www.geeksforgeeks.org/merge-sort/
218      */
219     private void sort(int[] arr, int l, int r) {
220         if (l < r)
221         {
222             // Find the middle point
223             int m = (l+r)/2;
224
225             // Sort first and second halves
226             sort(arr, l, m);
227             sort(arr, m+1, r);
228
229             // Merge the sorted halves
230             merge(arr, l, m, r);
231         }
232     }
233
234     /**
235      * Merge sort helper function
236      * taken from https://www.geeksforgeeks.org/merge-sort/
237      * Merges two subarrays of arr[].
238      * First subarray is arr[l..m]
239      * Second subarray is arr[m+1..r]
240      */
241     private void merge(int arr[], int l, int m, int r)
242     {
243         // Find sizes of two subarrays to be merged
244         int n1 = m - l + 1;
245         int n2 = r - m;
246
247         /* Create temp arrays */
248         int L[] = new int [n1];
249         int R[] = new int [n2];
250
251         /*Copy data to temp arrays*/
252         for (int i=0; i<n1; ++i)
253             L[i] = arr[l + i];
254         for (int j=0; j<n2; ++j)
255             R[j] = arr[m + 1+ j];
256
257
258         /* Merge the temp arrays */
259
260         // Initial indexes of first and second subarrays
261         int i = 0, j = 0;
262
263         // Initial index of merged subarray array
264         int k = l;
265         while (i < n1 && j < n2)
266         {
267             if (L[i] <= R[j])
268             {
269                 arr[k] = L[i];
270                 i++;
271             }

```

```

272         else
273         {
274             arr[k] = R[j];
275             j++;
276         }
277         k++;
278     }
279
280     /* Copy remaining elements of L[] if any */
281     while (i < n1)
282     {
283         arr[k] = L[i];
284         i++;
285         k++;
286     }
287
288     /* Copy remaining elements of R[] if any */
289     while (j < n2)
290     {
291         arr[k] = R[j];
292         j++;
293         k++;
294     }
295 }
296
297 @Override
298 public String toString() {
299     return "Merge Sort";
300 }
301 }
302
303 /**
304  * The class that represents bubble sort.
305  * Implements ISortStrategy with bubble sort.
306  * Implementation taken from https://www.geeksforgeeks.org/bubble-sort/
307  */
308 class BubbleSort implements ISortStrategy{
309
310     /**
311      * Implemenation of Bubble Sort
312      * taken from https://www.geeksforgeeks.org/bubble-sort/
313      */
314     @Override
315     public void sort(int[] arr) {
316         System.out.println("Bubble Sort executed.");
317         int n = arr.length;
318         for (int i = 0; i < n-1; i++)
319             for (int j = 0; j < n-i-1; j++)
320                 if (arr[j] > arr[j+1])
321                 {
322                     // swap temp and arr[i]
323                     int temp = arr[j];
324                     arr[j] = arr[j+1];
325                     arr[j+1] = temp;
326                 }
327     }
328
329     @Override
330     public String toString() {
331         return "Bubble Sort";
332     }
333 }

```