```java
package esof322hw2;

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

/**
 * ESOF322 HW 2 Client to choose math software
 * @author Karl Molina, Dana Parker
 */
public class Client {
    /**
     * Main method that lets you select a math software to sort numbers
     * with a default sort strategy
     * then keep that same math software but change the sort strategy
     * @param args
     */
    public static void main(String[] args) {
        //numbers to sort
        int[] numbers = {9, 3, 6, 8, 7};
        Scanner in = new Scanner(System.in);

        //mathSoftware that holds the sorting strategies
        MathSoftware mathSoftware = null;
        System.out.println("You must sort these numbers: " + Arrays.toString(numbers));
        System.out.println("Which math software would you like to use to sort them?");
        System.out.println("Mathematica (1), MTool (2), or MyMath (3)?");

        //input chooses which software to use
        switch (in.nextInt()) {
            case 1:
                mathSoftware = new Mathematica();
                break;
            case 2:
                mathSoftware = new MTool();
                break;
            case 3:
                mathSoftware = new MyMath();
                break;
            default:
                System.out.println("You have entered an invalid number. Program
                terminated.");
                in.close();
                return;
        }

        System.out.println("Executing " + mathSoftware + "'s default sorting
        algorithm...");
        mathSoftware.mathSort(numbers);
        System.out.println("Your result is: " + Arrays.toString(numbers));

        //new numbers to sort
        numbers = new int[] {5, 3, 1, 7, 2};
        System.out.println("\nYou must now sort these numbers: " +
        Arrays.toString(numbers));
        System.out.println("Which new sorting algorithm would you like to use?");
        System.out.println("InsertionSort (1), MergeSort (2), or BubbleSort (3)?");

        //input chooses which sort strategy to switch to
        switch (in.nextInt()) {
            case 1:
                mathSoftware.setSortStrategy(new InsertionSort());
                break;
            case 2:
                mathSoftware.setSortStrategy(new MergeSort());
                break;
            case 3:
                mathSoftware.setSortStrategy(new BubbleSort());
                break;
            default:
```

```
67                  System.out.println("You have entered an invalid number. Program
                    terminated.");
68                  in.close();
69                  return;
70          }
71          in.close();
72
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
            algorithm to " + sortStrategy);
95          this.sortStrategy = sortStrategy;
96      }
97
98      public void mathSort(int[] input) {
99          System.out.println("Sorting the numbers using " + sortStrategy);
100         sortStrategy.sort(input);
101     }
102 }
103
104 /**
105  * A math software that has the default sort strategy of insertion sort.
106  */
107 class Mathematica extends MathSoftware {
108
109     public Mathematica() {
110         super(new InsertionSort());
111         System.out.println("\tInitializing Mathematica");
112     }
113
114     @Override
115     public String toString() {
116         return "Mathematica";
117     }
118 }
119
120 /**
121  * A math software that has the default sort strategy of merge sort.
122  */
123 class MTool extends MathSoftware {
124
125     public MTool() {
126         super(new MergeSort());
127         System.out.println("\tInitializing MTool");
128     }
129
130     @Override
131     public String toString() {
132         return "MTool";
133     }
```

```
}

/**
 * A math software that has the default sort strategy of bubble sort.
 */
class MyMath extends MathSoftware {

    public MyMath() {
        super(new BubbleSort());
        System.out.println("\tInitializing MyMath");
    }

    @Override
    public String toString() {
        return "MyMath";
    }
}

/**
 * Interface that defines the method sort().
 */
interface ISortStrategy {

    public void sort(int[] input);
}

/**
 * The class that represents insertion sort.
 * Implements ISortStrategy with insertion sort.
 * Implementation from https://www.geeksforgeeks.org/insertion-sort/
 */
class InsertionSort implements ISortStrategy {

    /**
     * Implementation of Insertion Sort taken from
     * https://www.geeksforgeeks.org/insertion-sort/
     */
    @Override
    public void sort(int[] arr) {
        System.out.println("Insertion Sort executed.");
        int n = arr.length;
        for (int i=1; i<n; ++i)
        {
            int key = arr[i];
            int j = i-1;

            /* Move elements of arr[0..i-1], that are
               greater than key, to one position ahead
               of their current position */
            while (j>=0 && arr[j] > key)
            {
                arr[j+1] = arr[j];
                j = j-1;
            }
            arr[j+1] = key;
        }
    }

    @Override
    public String toString() {
        return "Insertion Sort";
    }
}

/**
 * The class that represents merge sort.
 * Implements ISortStrategy with merge sort.
 * Implementation from https://www.geeksforgeeks.org/merge-sort/
 */
```

```
class MergeSort implements ISortStrategy {

    /**
     * Sort function that uses the help merge sort functions
     */
    @Override
    public void sort(int[] input) {
        System.out.println("Merge Sort executed.");
        sort(input, 0, input.length-1);
    }

    /**
     * Sort helper function
     * Implementation of merge sort
     * taken from https://www.geeksforgeeks.org/merge-sort/
     */
    private void sort(int[] arr, int l, int r) {
        if (l < r)
        {
            // Find the middle point
            int m = (l+r)/2;

            // Sort first and second halves
            sort(arr, l, m);
            sort(arr , m+1, r);

            // Merge the sorted halves
            merge(arr, l, m, r);
        }
    }

    /**
     * Merge sort helper function
     * taken from https://www.geeksforgeeks.org/merge-sort/
     * Merges two subarrays of arr[].
     * First subarray is arr[l..m]
     * Second subarray is arr[m+1..r]
     */
    private void merge(int arr[], int l, int m, int r)
    {
        // Find sizes of two subarrays to be merged
        int n1 = m - l + 1;
        int n2 = r - m;

        /* Create temp arrays */
        int L[] = new int [n1];
        int R[] = new int [n2];

        /*Copy data to temp arrays*/
        for (int i=0; i<n1; ++i)
            L[i] = arr[l + i];
        for (int j=0; j<n2; ++j)
            R[j] = arr[m + 1+ j];


        /* Merge the temp arrays */

        // Initial indexes of first and second subarrays
        int i = 0, j = 0;

        // Initial index of merged subarry array
        int k = l;
        while (i < n1 && j < n2)
        {
            if (L[i] <= R[j])
            {
                arr[k] = L[i];
                i++;
            }
```

```
            else
            {
                arr[k] = R[j];
                j++;
            }
            k++;
        }

        /* Copy remaining elements of L[] if any */
        while (i < n1)
        {
            arr[k] = L[i];
            i++;
            k++;
        }

        /* Copy remaining elements of R[] if any */
        while (j < n2)
        {
            arr[k] = R[j];
            j++;
            k++;
        }
    }

    @Override
    public String toString() {
        return "Merge Sort";
    }
}

/**
 * The class that represents bubble sort.
 * Implements ISortStrategy with bubble sort.
 * Implementation taken from https://www.geeksforgeeks.org/bubble-sort/
 */
class BubbleSort implements ISortStrategy{

    /**
     * Implemenation of Bubble Sort
     * taken from https://www.geeksforgeeks.org/bubble-sort/
     */
    @Override
    public void sort(int[] arr) {
        System.out.println("Bubble Sort executed.");
        int n = arr.length;
        for (int i = 0; i < n-1; i++)
            for (int j = 0; j < n-i-1; j++)
                if (arr[j] > arr[j+1])
                {
                    // swap temp and arr[i]
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
    }

    @Override
    public String toString() {
        return "Bubble Sort";
    }
}
```