```
/*
 *Kevin Yu
 *1/12/22
 *This takes input and sorts it using differnt methods.
 */

package sort;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class sort {

	Scanner consoleInput = new Scanner(System.in);
	String input;
	Scanner fileInput;
	int[] inputArray;
	long startTime;

	public sort() {
		System.out.println("Enter a number for the input file.");
		System.out.println("1: input.txt	2: input2.txt	3: input3.txt
	4: input4.txt");
		input = consoleInput.nextLine();
		if (input.length() != 1 && input.charAt(0) != '1' && input.charAt(0) !=
'2'
				&& input.charAt(0) != '3' && input.charAt(0) != '4')
{
			System.out.println("Enter 1, 2, 3, or 4.");
			while (input.length() != 1 && input.charAt(0) != '1' &&
input.charAt(0) != '2'
				&& input.charAt(0) != '3' && input.charAt(0) != '4')
{
				input = consoleInput.nextLine();
			}
		}
		try {
			fileInput = new Scanner(new File("input" + input.charAt(0) +
".txt"));
		} catch (FileNotFoundException ex) {
			ex.printStackTrace();
			System.exit(0);
		}
		String infile = fileInput.nextLine();
		String[] inputStringArray = infile.split(",");
		inputArray = new int[inputStringArray.length];
		for (int i = 0; i < inputStringArray.length; i++) {
			inputArray[i] = Integer.parseInt(inputStringArray[i]);
			System.out.println(inputArray[i]);
		}
		System.out.println("Enter a number for the sort you want to use.");
		System.out.println("1: Bubble 2: Selection	3: Table	4: Quick");
		input = consoleInput.nextLine();
		if (input.length() != 1 && input.charAt(0) != '1' && input.charAt(0) !=
'2'
				&& input.charAt(0) != '3') {
```

```
                  System.out.println("Enter 1, 2, 3, or 4.");
                  while (input.length() != 1 && input.charAt(0) != '1' &&
input.charAt(0) != '2'
                              && input.charAt(0) != '3' && input.charAt(0) != '4')
{
                        input = consoleInput.nextLine();
                  }
            }
            startTime = System.currentTimeMillis();
            if (input.equals("1")) {
                  inputArray = bubbleSort(inputArray);
            }
            if (input.equals("2")) {
                  inputArray = selectionSort(inputArray);
            }
            if (input.equals("3")) {
                  inputArray = tableSort(inputArray);
            }
            if (input.equals("4")) {
                  inputArray = quickSort(inputArray, 0, (inputArray.length)-1);
            }
            long totalTime = System.currentTimeMillis() - startTime;
            PrintWriter pw;
            try {
                  pw = new PrintWriter(new FileWriter(new File("output.txt")));
                  String output = "";
                  for (int i = 0; i < inputArray.length; i++) {
                        output += inputArray[i] + ",";
                  }
                  output += "\nTotal Time:" + totalTime;
                  pw.write(output);
                  pw.close();
            } catch (IOException ex) {
                  ex.printStackTrace();
                  System.exit(0);
            }

      }

      //compare each pair of numbers and move the larger to the right
      int[] bubbleSort(int[] array) {
            for (int j = 0; j < array.length; j++) {
                  for (int i = 0; i < array.length - 1; i++) {
                        //if the one on the left is larger
                        if (array[i] > array[i+1]) {
                              //swap!
                              int temp = array[i];
                              array[i] = array[i+1];
                              array[i+1] = temp;
                        }
                  }
            }
            return array;
      }
      //find the smallest and move it to the front
      int[] selectionSort(int[] array) {
            for (int j = 0; j < array.length; j++) {
                  int smallestNumber = array[j];
                  int smallestIndex = j;
```

```
                for (int i = j; i < array.length; i++) {
                      if (array[i] < smallestNumber) {
                            smallestNumber = array[i];
                            smallestIndex = i;
                      }
                }
                int temp = array[smallestIndex];
                array[smallestIndex] = array[j];
                array[j] = temp;
          }

          return array;
     }

     //Count how often you see each number, print out  the number of times
     int[] tableSort(int[] array) {
          int[] tally = new int[1001];
          for (int i = 0; i < array.length; i++) {
                tally[array[i]]++;
          }

          int count = 0;
          //i keeps track of the actual number
          for (int i = 0; i < tally.length; i++) {
                //j keeps track of how many times we've seen that number
                for (int j = 0; j < tally[i]; j++) {
                      array[count] = i;
                      count++;

                }
          }
          return array;
     }

     static void swap (int[] array, int start, int end) {
          int tmp = array[start];
          array[start] = array[end];
          array[end] = tmp;
     }

     static int partition(int[] array, int start, int end) {
          //Sets Pivot
          int pivot = array[end];

          //Sets the threshold value
          int i = (start - 1);

          //The counter value that goes through the array and finds the value to
swap
          for (int j = start; j <= end - 1; j++) {
                //if current value is smaller then the pivot, it swaps it with
the threshold value
                if (array[j] < pivot) {
                      i++;
                      swap(array, i, j);
                }
          }
          swap(array, i+1, end);
          return (i+1);
```

```
	}


	int[] quickSort(int[] array, int start, int end) {
		if (start < end) {
			//finds a new pivot
			int pivot = partition(array, start, end);
			//resorts the array with a new pivot every time until it is
finished
			quickSort(array, start, pivot -1);
			quickSort(array, pivot + 1, end);
		}
		return array;
	}

	public static void main(String[] args) {
		new sort();
	}
}
```