Programming Assignment 3

Wrapper Classes, Text Processing, and Recursion with Merge Sort

Due: March 12

<Objective>

To demonstrate an understanding of recursive methods, text processing, and boxed primitives.

<Requirements>

You will implement a Java application that sorts and prints floating-point values read from a text file.

The application determines the name of the text file from the input provided by the customer through the keyboard. If the user supplies no arguments, or if the user supplies two or more arguments, the application will output “Invalid command-line arguments” and then keep asking the correct file name

```
$ java YiZhu3
Please input the file which contains numbers you want to sort
---->
Invalid command-line arguments.
Please input the file which contains numbers you want to sort
---->file1.txt file2.txt file3.txt
Invalid command-line arguments.
```

The application loops over every line of the input file. For every iteration, the application parses the line as a Double. If a line from the file is empty, the application terminates after writing to Standard Error, “Empty line encountered.”.

```
$ cat bad-input-1.txt
1.0
```

```
3.0
4.0
5.0
$ java YiZhu3
Please input the file which contains numbers you want to sort
---->bad-input-1.txt
Empty line encountered.
```

If a line from the input file contains anything other than one valid Double, the application terminates after writing to Standard Error, “Invalid line encountered: 'XXX'”, where 'XXX' is the contents of the first invalid line.

```
$ cat bad-input-2.txt
1.0
two
3.0 4.0
four
5.0

$ java YiZhu3
Please input the file which contains numbers you want to sort
---->bad-input-2.txt
Invalid line encountered: 'two'.
```

If any other I/O error occurs while reading the input file, the application terminates after writing to Standard Error, “Failed to read input file: XXX”.

```
$ java YiZhu3
Please input the file which contains numbers you want to sort
---->.
Failed to read input file: .
```

The application stores the parsed values from the text file into a collection of type ArrayList<Double>. Once all values are added to this collection, the application creates an array of type double[] that is the same length as the size of

the collection. The application "unboxes" the values from the collection and stores the corresponding primitive values into the array.

The application sorts the array using the recursive MergeSort algorithm (See the pseudo code below). You can implement the merge sort algorithm based on the following pseudo code and implementation hint or you may follow other materials (e.g. the merge sort algorithm in your discrete math textbook).

If you use the other materials for merge sort, please indicate the materials you use when you submit the project. If you do not mention with your submission, I assume that you implement based on the following description.

```
procedure MERGESORT(V, S, min, max) ≡
var
        V : an array of values
        S : a scratch array of the same type and size as V
        min : a minimum index to sort in V, inclusive
        max : a maximum index to sort in V, inclusive
begin
1.    if min = max then return
2.      lhsMax ← ⌊(min + max) / 2⌋
3.      rhsMin ← lhsMax + 1
4.      MERGESORT(V, S, min, lhsMax)
5.      MERGESORT(V, S, rhsMin, max)
6.      S[min … max] ← V[min … max]
7.      lhs ← min
8.      rhs ← rhsMin
9.      dst ← min
10.   while lhs ≤ lhsMax and rhs ≤ max do
11.     if S[lhs] < S[rhs] then do
12.       V[dst] ← S[lhs]
13.       lhs ← lhs + 1
14.     else
15.       V[dst] ← S[rhs]
16.       rhs ← rhs + 1
17.     end if
18.       dst ← dst + 1
19.   end while
20.   while lhs ≤ lhsMax do
21.     V[dst] ← S[lhs]
22.     lhs ← lhs + 1
23.     dst ← dst + 1
24.   end while
25.   while rhs ≤ max do
```

```
26.     V[dst] ← S[rhs]
27.     rhs ← rhs + 1
28.     dst ← dst + 1
29.   end while
end
```

The implementation defines a method with the following signature to start the sort:

```
static void mergeSort(double[] values)
```

The `mergeSort(double[])` method allocates a scratch array that is the same size as the input array. This method then calls a method with the following signature to perform the recursive portion of the algorithm:

```
static void mergeSort(
  double[] values,
  double[] scratch,
  int min,
  int max)
```

Once the sort finishes, the application writes the contents of the sorted array to Standard Output. Each value in the array is printed consecutively as lines of output. When all values have been printed, the application terminates immediately without displaying any additional information.

```
Suppose you have created "list-5.txt" and saved in the disk and the content is as
follows:
2.8
1.9
4.6
5.5
3.7

$ java YiZhu3
Please input the file which contains numbers you want to sort
----> list-5.txt
1.9
2.8
3.7
4.6
```

```
5.5
```

<Example of Output>

```
Suppose you have created "sample-input-10.txt" and saved in the disk. The content are
as follows:
0.6278543339859269
0.6046406161572284
0.7167025618977285
0.2842196624797939
0.3811228959311541
-0.40900050358949513
0.2945547593587171
0.37008144080538685
0.8000032729432853
0.49645681304284206

$ java YiZhu3
Please input the file which contains numbers you want to sort
----> sample-input-10.txt
-0.40900050358949513
0.2842196624797939
0.2945547593587171
0.37008144080538685
0.3811228959311541
0.49645681304284206
0.6046406161572284
0.6278543339859269
0.7167025618977285
0.8000032729432853
```

<Resources>

- Merge Sort Description and Animation
- Relevant API
  1. ArrayList

2. Double
3. Scanner