

Chapter 2 Programming

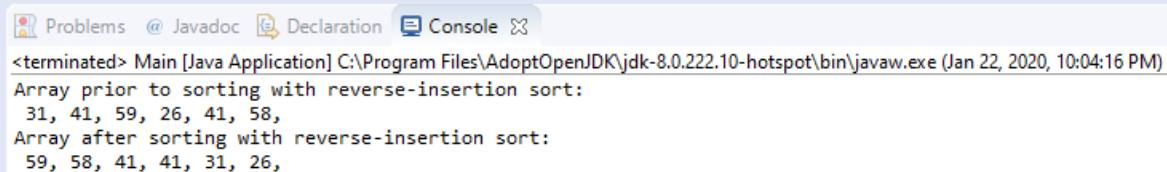
Insertion Sort:

Here I have included my results of insertion sort on the array < 31, 41, 50, 26, 41, 58 >. This code works by placing an element in a 'key' temporary position, then finding where it fits in an already sorted array in ascending order.

```
<terminated> Main [Java Application] C:\Program Files\AdoptOpenJDK\bin\javaw.exe (Jan 22, 2020, 10:04:16 PM)
Array prior to sorting with insertion sort:
31, 41, 59, 26, 41, 58,
Array after sorting in insertion sort:
26, 31, 41, 41, 58, 59,
```

Reverse-Insertion Sort:

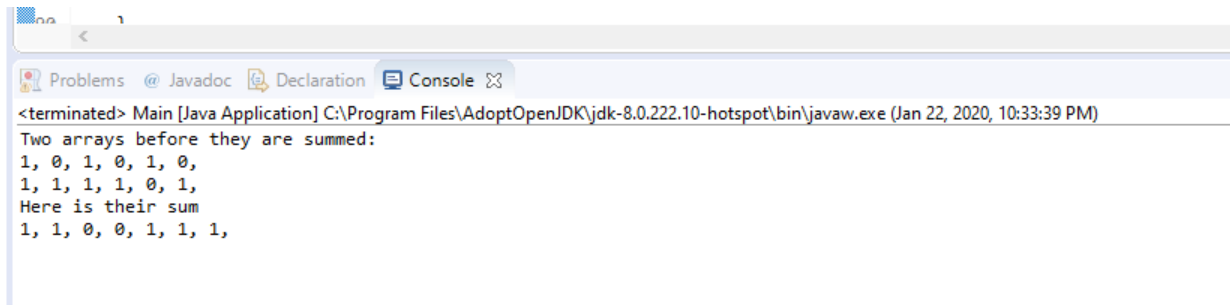
Here I have included my results of the reverse-insertion sort on that same array. This algorithm works very similar to the Insertion Sort, except that it just places them in the descending order by reordering only one line of code.



```
<terminated> Main [Java Application] C:\Program Files\AdoptOpenJDK\bin\javaw.exe (Jan 22, 2020, 10:04:16 PM)
Array prior to sorting with reverse-insertion sort:
31, 41, 59, 26, 41, 58,
Array after sorting with reverse-insertion sort:
59, 58, 41, 41, 31, 26,
```

Binary Array Addition:

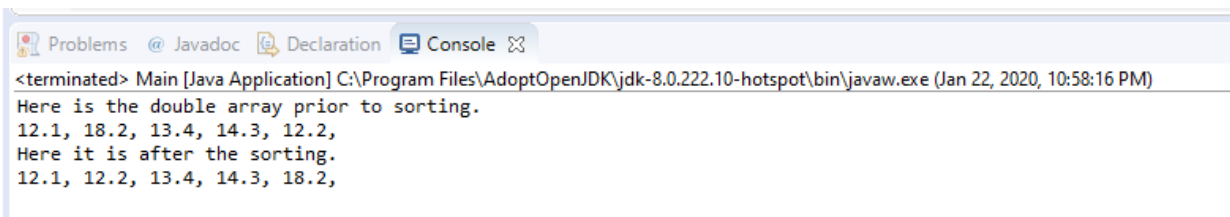
Here I have worked up the programming of adding two binary arrays, and shows them here. This code is done by a statement of if conditions, when the values of the previous two binary arrays are summed together. If the sum is greater or equal to 2, then it is incremented to the next base of two.



```
na 1
Problems @ Javadoc Declaration Console
<terminated> Main [Java Application] C:\Program Files\AdoptOpenJDK\jdk-8.0.222.10-hotspot\bin\javaw.exe (Jan 22, 2020, 10:33:39 PM)
Two arrays before they are summed:
1, 0, 1, 0, 1, 0,
1, 1, 1, 1, 0, 1,
Here is their sum
1, 1, 0, 0, 1, 1, 1,
```

MergeSort and Merge

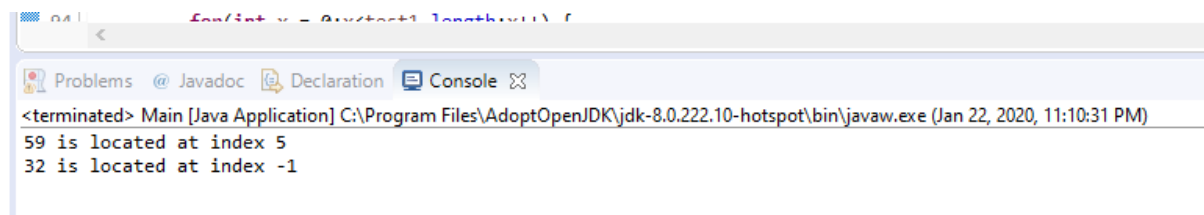
Here is the coding for the Merge and Merge-Sort algorithms line for line translation. This code is a very effective recursive method to sorting. The base case is arrived at by splitting the array into sub-array's until it is of length 1, which is sorted; then merges the already sorted piles, by checking which of the piles contains the lowest; then uses sentinel values to prevent the code from ending early in the merging portion.



```
Problems @ Javadoc Declaration Console
<terminated> Main [Java Application] C:\Program Files\AdoptOpenJDK\jdk-8.0.222.10-hotspot\bin\javaw.exe (Jan 22, 2020, 10:58:16 PM)
Here is the double array prior to sorting.
12.1, 18.2, 13.4, 14.3, 12.2,
Here it is after the sorting.
12.1, 12.2, 13.4, 14.3, 18.2,
```

Binary Search

Here is my Binary Search Results from looking for two numbers in the first array used. This code uses the effectiveness of logarithms, by attempting to remove half of the number of elements each time it is run. There is an anchor that checks against the half-way difference from the previously determined section of indexes. This particular code will return the index number; if that number is inside of that particular sorted array; and will provide a -1 when that number is not located in that array.



```
na 1
Problems @ Javadoc Declaration Console
<terminated> Main [Java Application] C:\Program Files\AdoptOpenJDK\jdk-8.0.222.10-hotspot\bin\javaw.exe (Jan 22, 2020, 11:10:31 PM)
59 is located at index 5
32 is located at index -1
```

BubbleSort

Bubble sort results on the very first array is provided. This method is a very rudimentary method of sorting; it has a terrible $O(n^2)$ worst case because this code takes a look at every element almost guaranteed twice; as it pairs each element with its neighbor; then slowly moves the largest to the highest index; then moves to find the second highest beginning again.



```
Problems @ Javadoc Declaration Console
<terminated> Main [Java Application] C:\Program Files\AdoptOpenJDK\jdk-8.0.222.10-hotspot\bin\javaw.exe (Jan 22, 2020, 11:13:43 PM)
Here is the array before it is sorted by bubblesort.
31, 41, 59, 26, 41, 58,
Here is the array after it is sorted.
26, 31, 41, 41, 58, 59,
```