**PARALLEL ALGORITHMS**

**ASSIGNMENT 3**

**TEAM #3 :** Sneha Didigam

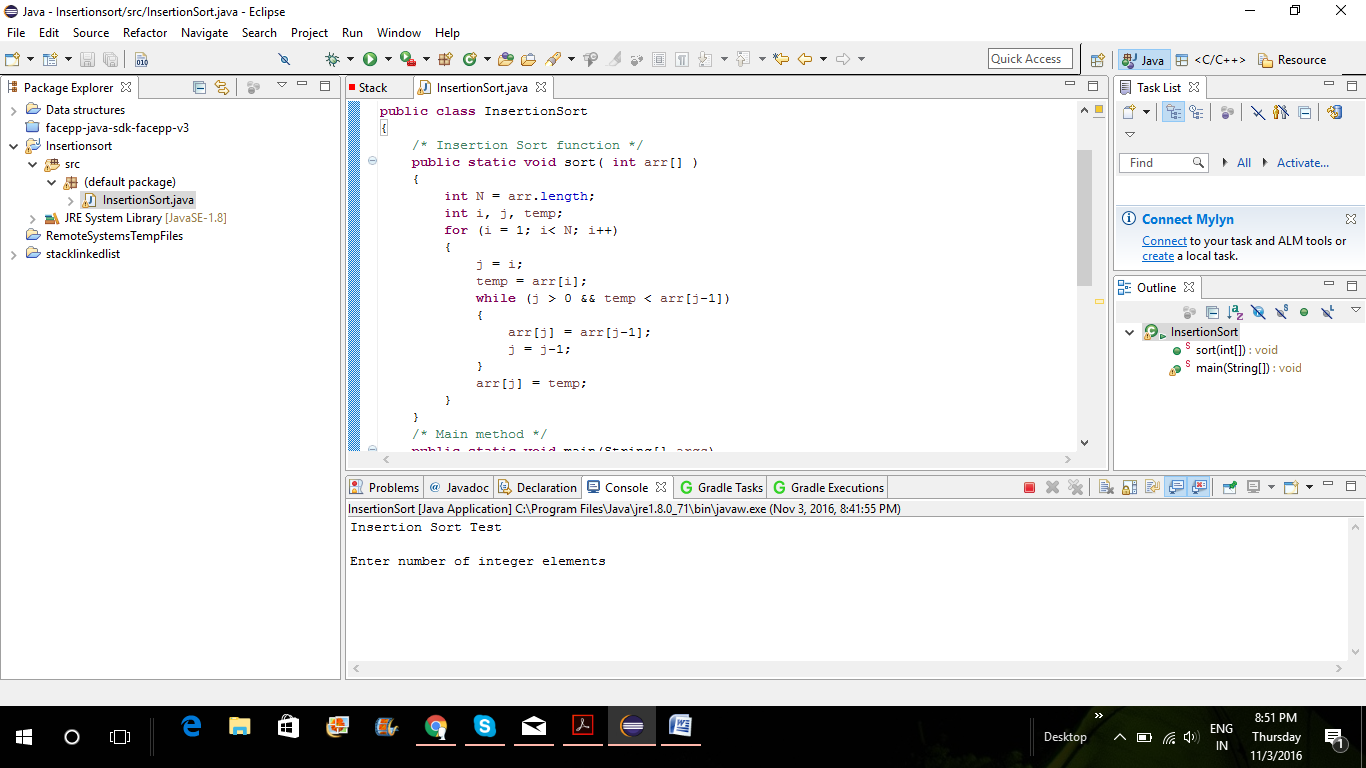
Depthi Penmetsa

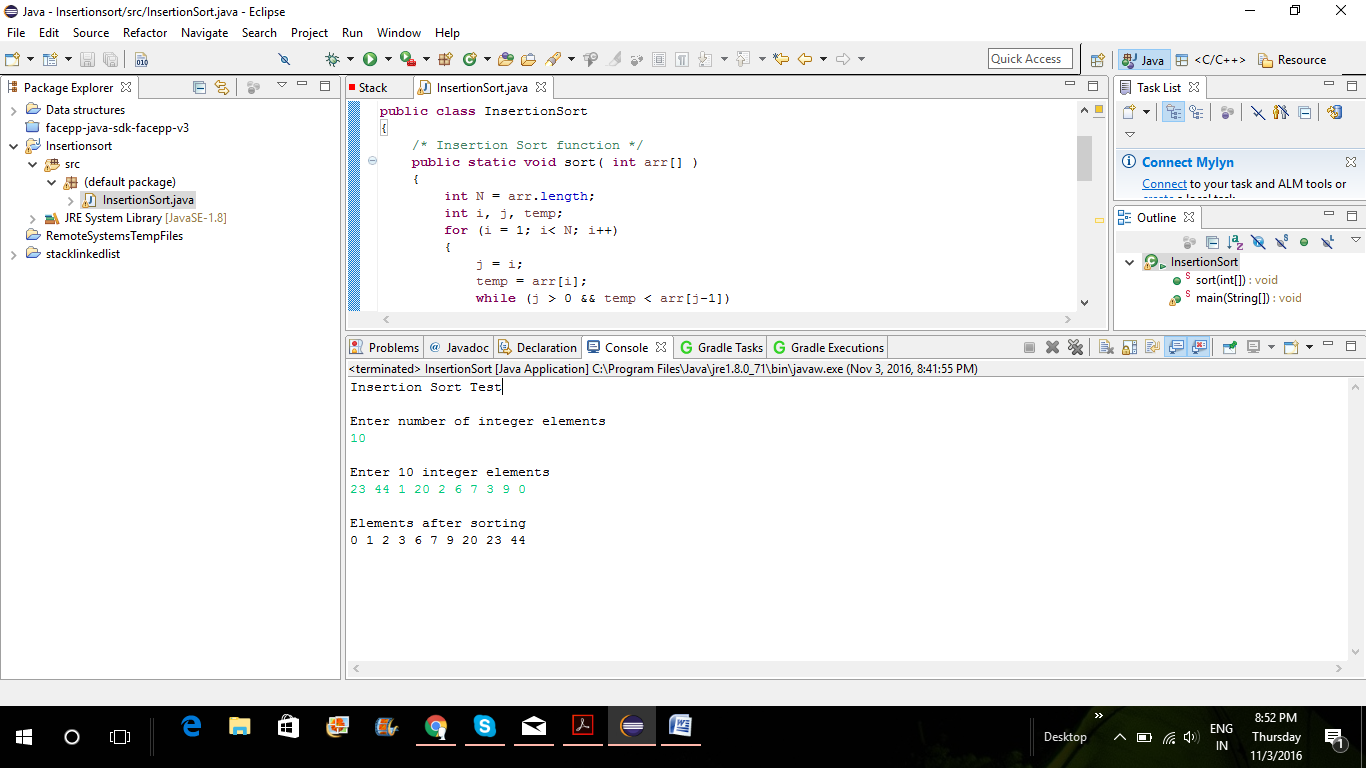
Sirisha Manthramurthy

**Insertion sort:**

1. */\**
2. *\* Java Program to Implement Insertion Sort*
3. *\*/*
5. **import** java.util.Scanner;
7. */\* Class InsertionSort \*/*
8. **public** **class** InsertionSort
9. {
10. */\* Insertion Sort function \*/*
11. **public** **static** **void** sort( **int** arr[] )
12. {
13. **int** N = arr.length;
14. **int** i, j, temp;
15. **for** (i = 1; i< N; i++)
16. {
17. j = i;
18. temp = arr[i];
19. **while** (j > 0 && temp < arr[j-1])
20. {
21. arr[j] = arr[j-1];
22. j = j-1;
23. }
24. arr[j] = temp;
25. }
26. }
27. */\* Main method \*/*
28. **public** **static** **void** main(String[] args)
29. {
30. Scanner scan = **new** Scanner( System.in );
31. System.out.println("Insertion Sort Test**\n**");
32. **int** n, i;
33. */\* Accept number of elements \*/*
34. System.out.println("Enter number of integer elements");
35. n = scan.nextInt();
36. */\* Create integer array on n elements \*/*
37. **int** arr[] = **new** **int**[ n ];
38. */\* Accept elements \*/*
39. System.out.println("**\n**Enter "+ n +" integer elements");
40. **for** (i = 0; i < n; i++)
41. arr[i] = scan.nextInt();
42. */\* Call method sort \*/*
43. sort(arr);
44. */\* Print sorted Array \*/*
45. System.out.println("**\n**Elements after sorting ");
46. **for** (i = 0; i < n; i++)
47. System.out.print(arr[i]+" ");
48. System.out.println();
49. }
50. }

**Screen shots:**

****

****

**Pseudo Code for Insertion sorting Technique in parallel:**

package com.umkc.insertion.sorting;

public class MyInsertionSort {

int[] a = {10,34,2,56,7,67,88,42};

if (a[] < 2)

return a

else

head = a[0];

rest = sorted list taken from (dropped(a,1));

i = count({x < head : x in rest});

in take(rest,i)++[head]++drop(rest,i);

}

}

**Output:**

2,7,10,34,42,56,67,88

**Time Complexity in parallel** : O(n log n)

**Comparison with other techniques**

1. **Selection sort** : Time complexity of selection sort is O(n^2), while parallel insertion sort

has complexity of O(n log n), insertion sort is better than selection sort.

2. **Counting sort** : Time complexity of counting sort is O(n^2), while parallel insertion sort

has complexity of O(n log n), insertion sort is better than counting sort.

3. **Merge sort** : Time complexity of merge sort is O(n log n), while parallel insertion sort has

complexity of O(n log n). Both Merge sort and insertion sort are having same time

complexity.

4. **Bubble sort** : Time complexity of bubble sort is O(n log n), while parallel insertion sort has

complexity of O(n log n). Both bubble and insertion sort are having same time complexity.

5. **Quick sort** : Time complexity of quick sort is O(n log n), while parallel insertion sort has

complexity of O(n log n). Both quick and quick sort are having same time complexity.