Insertion Sort

Best	Average	Worst
$O(n)$	$O(n^2)$	$O(n^2)$

The Insertion Sort uses an **incremental** approach. It works similar to the way you might organize a hand of cards. The unsorted cards begin face down on the table and are picked up one by one. As each new unsorted card is picked up, it is inserted into the correct order in your organized hand of cards.

The following shows the sequence of steps in an Insertion Sort:

2	**5**	1	4	3	Original data. The 2 is our "hand" so insert the 5 into it.
2	5	1	4	3	**End of 1st pass**: The 5 is already in the right place. No need to move.
2	5	**1**	4	3	Our "hand" is now 2, 5. Think of inserting the 1 into it.
1	2	5	4	3	**End of 2nd pass:** Notice the 1 has been **inserted** in the right place.
1	2	5	**4**	3	Our "hand" is now 1, 2, 5. Think of inserting the 4 into it.
1	2	4	5	3	**End of 3rd pass:** Notice the 4 has been **inserted** in the right place.
1	2	4	5	**3**	Our "hand" is now 1, 2, 4, 5. Think of inserting the 3 into it.
1	2	3	4	5	**End of 4th pass:** Notice the 3 has been **inserted** in the right place.

```
public static void sort(int a[ ] ) { //This will do an ascending sort
        int itemToInsert, j;
        boolean keepGoing;
        //On kth pass, insert item k into its correct position among the first k items in the array
        for(int k = 1; k < a.length; k++)
        {
                //Go backwards through the list, looking for the slot to insert a[k]
                itemToInsert = a[k];
                j = k –1;
                keepGoing = true;

                while((j >= 0) && keepGoing)
                {
                        if (itemToInsert < a[j] )
                        {
                                a[j + 1] = a[j]; //Salient feature
                                j--;
                                if(j == -1) //special case for inserting an item at [0]
                                a[0] = itemToInsert;
                        }
                        else //Upon leaving loop, j + 1 is the index where itemToInsert belongs
                        {
                            keepGoing = false;
                            a[j + 1] = itemToInsert;
                        }
                    }
                }
            }
```

An advantage:
The Insertion Sort has an advantage over the Selection Sort since it takes advantage of a partially ordered list. This is evidenced by the fact that in a best case, big O for an Insertion Sort is O(n), whereas for a Selection Sort, it is always O(n^2).