* To write a program which can sort any data type of input implement the comparable interface which has the compareTo() method.
* Every datatype has its respective compareTo() method.
* compareTo() throws an null exception for incompatible types
* The algorithm follows that for the every iteration I find the minimum entry remaining and swap it with the ‘i’ position.
* Thus, it does quadratic number of comparisons and linear number of exchanges irrespective of whether the array of sorted or not.
* In this type of sorting in the I iteration the element at a[i] is exchanged with all the elements towards its left which are smaller itself.
* Insertion sort is twice as fast as selection sort.
* First sort the array by h sort then by g sort and soon.
* There are various studies on which increment process to follow. One of the good sorting increments is kanou’s 3x+ 1 increment.
  + Eg: 1,4,13,40,121,….
* It takes NlogN time for solving.
* It makes use of the divide and conquer type techniques.
* Thus the array is divided to the base case merged and so on makes use of recursion to finally yield the result.
* Bottom up merge sort can also be used to sort an array. It doesn’t require recursion to sort an array.
* Analysis of running time is done using decision trees.
* For smaller length problems insertion sort should be used instead of merge sort as it is less complicated.
* Merge sort requires extra space for the auxiliary array. Thus, a merge of only half size of that of the total memory available can be implemented.
* Comparator class objects are used to sort an table by different sorting keys.
* One advantage of quick sort over merge sort is that it does not require the additional array.
* It still has O(NlogN).
* Shuffling is required to guarantee performance.
* Quicksort is faster than mergesort.
* Some ways to make it even faster:
  + Cut it off to use insertion sort for 10 items or so.
* **Quick Select** is a selection algorithm which in general takes linear time on average.
* Stable sorts are the sorts which preserve the sorting order of the first sorting key and then sort the array according to the second sorting key.
* **Insertion sort and merge sort** are **stable** but **not** **quick sort, selection sort and shell sort**.
* But in general it depends whether a sort is stable or not based on how it is coded.
* Also if a sorting technique has long distance sorting it may not be stable as it passes the key across many keys and may miss something.
* Can be imported by import.java.util.Arrays.
* It uses mergesort for objects and tuned quicksort for primitives.
* Uses different technique for no concrete reasons. May be just because if someone is using objects space might not be an issue so merge sort can be used. Plus merge sort is stable and guarantees NlogN performance.
* Has a method which implements comparable for data types.
* Has a method that uses comparator.
* Is an in place sorting algorithm.
* It does the job in guaranteed O(NlogN) times.
* The trace is as follows:
* 
* Disadvantages:
  + Inner loops are longer than quicksort’s.
  + Makes poor use of cache memory.
  + Not Stable.
* 