Project Title: Insertion Sort Algorithm Implementation

Overview
This project implements the insertion sort algorithm in Java, designed to sort a sequence of strings provided through standard input. The primary goal is to demonstrate a fundamental sorting method that is easy to understand and implement.

Algorithm Description in Plain English

- **Initial Setup:**
  The algorithm begins by conceptually dividing the array into two parts: a sorted section and an unsorted section. Initially, the first element is treated as already sorted, while the rest of the array remains unsorted.

- **Iterative Insertion Process:**
  The algorithm then processes each element in the unsorted section one at a time. For each element, it compares the element with those in the sorted section.

  - Starting from the current element, it checks the elements to its left in the sorted section.
  - If the current element is smaller than any element in the sorted section, the larger elements are shifted one position to the right to create space.
  - The current element is then inserted into the correct position within the sorted section.

- **Repeating the Process:**
  This process is repeated for every element in the unsorted section until the entire array is sorted in ascending order.

Key Characteristics

- **Simplicity:**
  The method is intuitive and serves as an excellent introduction to sorting algorithms.

- **Stability:**
  Insertion sort maintains the relative order of equal elements, making it a stable sorting algorithm.

- **Adaptive Performance:**
  Although its worst-case performance is quadratic ($O(n^2)$), it performs very efficiently on nearly sorted arrays—often close to linear time.

- **In-Place Sorting:**
  The algorithm requires only a constant amount of extra memory because it sorts the array without needing additional data structures.

- **Flexibility:**
  Besides sorting based on the natural ordering of the elements, the algorithm can also sort

using a custom comparator, allowing for flexible ordering criteria.