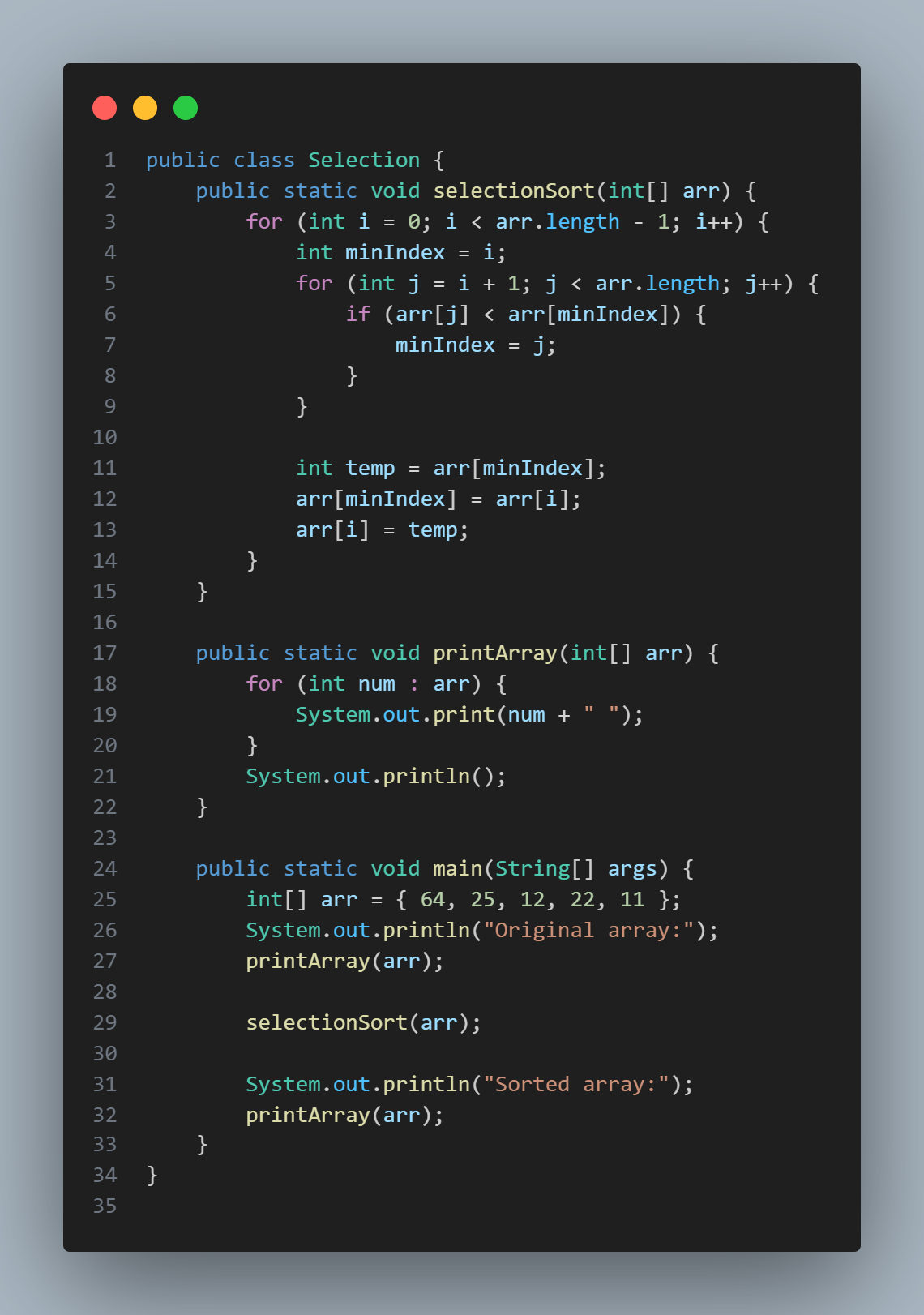
**Selection Sort**

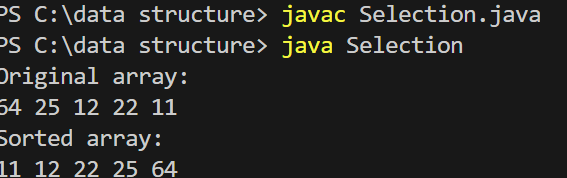
**Description: divides the input list into two parts: the sublist of items already sorted, which is built up from left to right, and the sublist of items remaining to be sorted that occupy the rest of the list. It repeatedly selects the smallest (or largest, depending on sorting order) element from the unsorted sublist, swapping it with the first unsorted element.**

**Advantages:**

* **Simple Implementation: Easy to understand and code.**
* **Memory Usage: Works in-place and requires only a small amount of additional memory.**
* **Simple to implement. Performs well on small lists and uses very little extra memory (O(1).**

**Application: Useful when memory write operations are costly since it does fewer writes.**

****

****

**Insertion Sort**

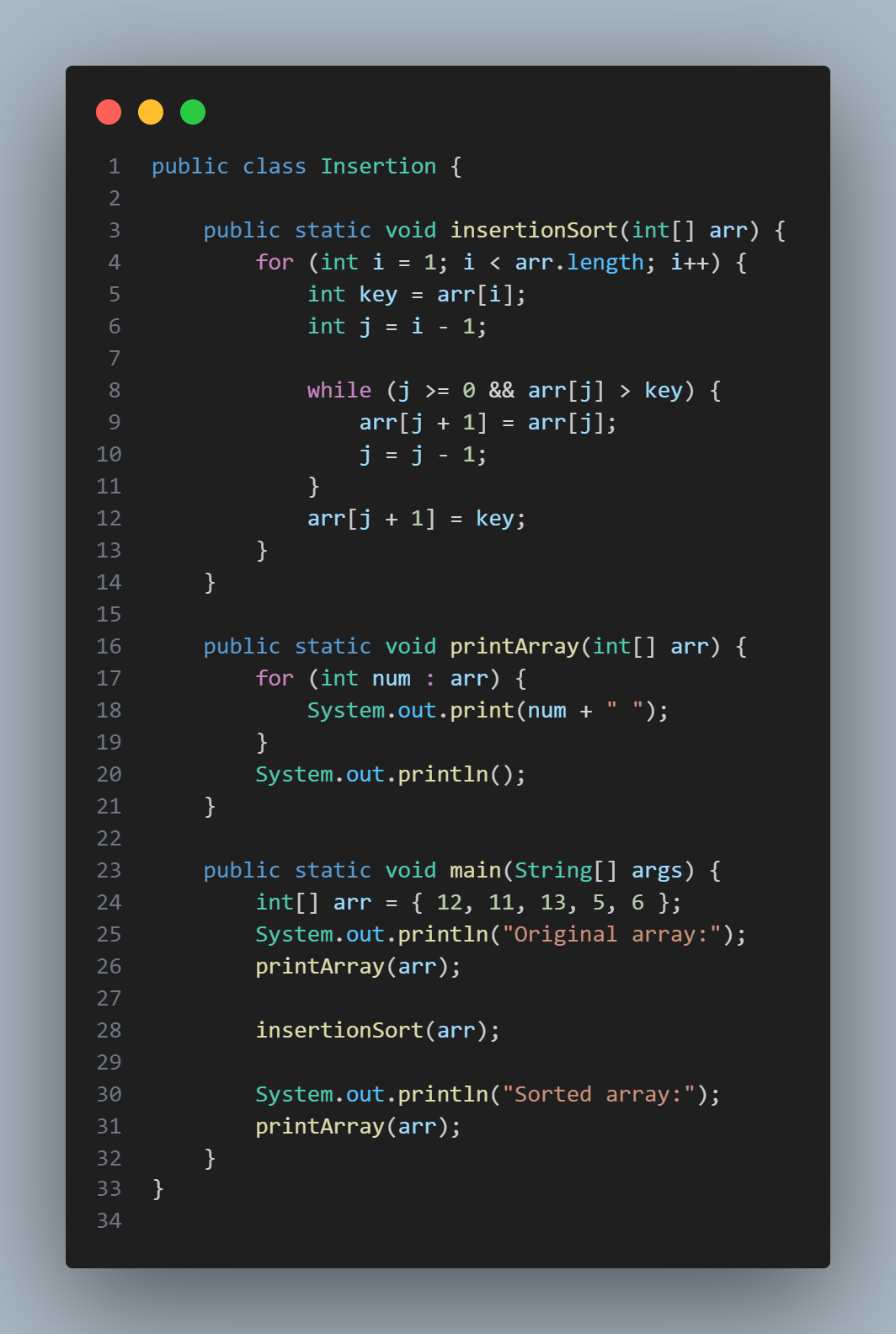
**Description: is a simple sorting algorithm that builds the final sorted array (or list) one item at a time. It’s much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort.**

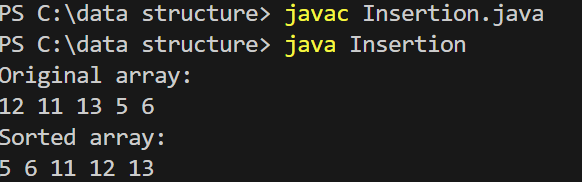
**works by iterating through an array one element at a time, inserting each element into its proper position within the previously sorted portion of the array.**

**Advantages:**

* **Simplicity: Easy to understand and implement.**
* **Adaptive: Efficient for data that is already substantially sorted.**
* **Stable: Maintains the relative order of records with equal keys.**
* **In-Place:**

**Application: Suitable for small datasets or nearly sorted arrays.**

****

****

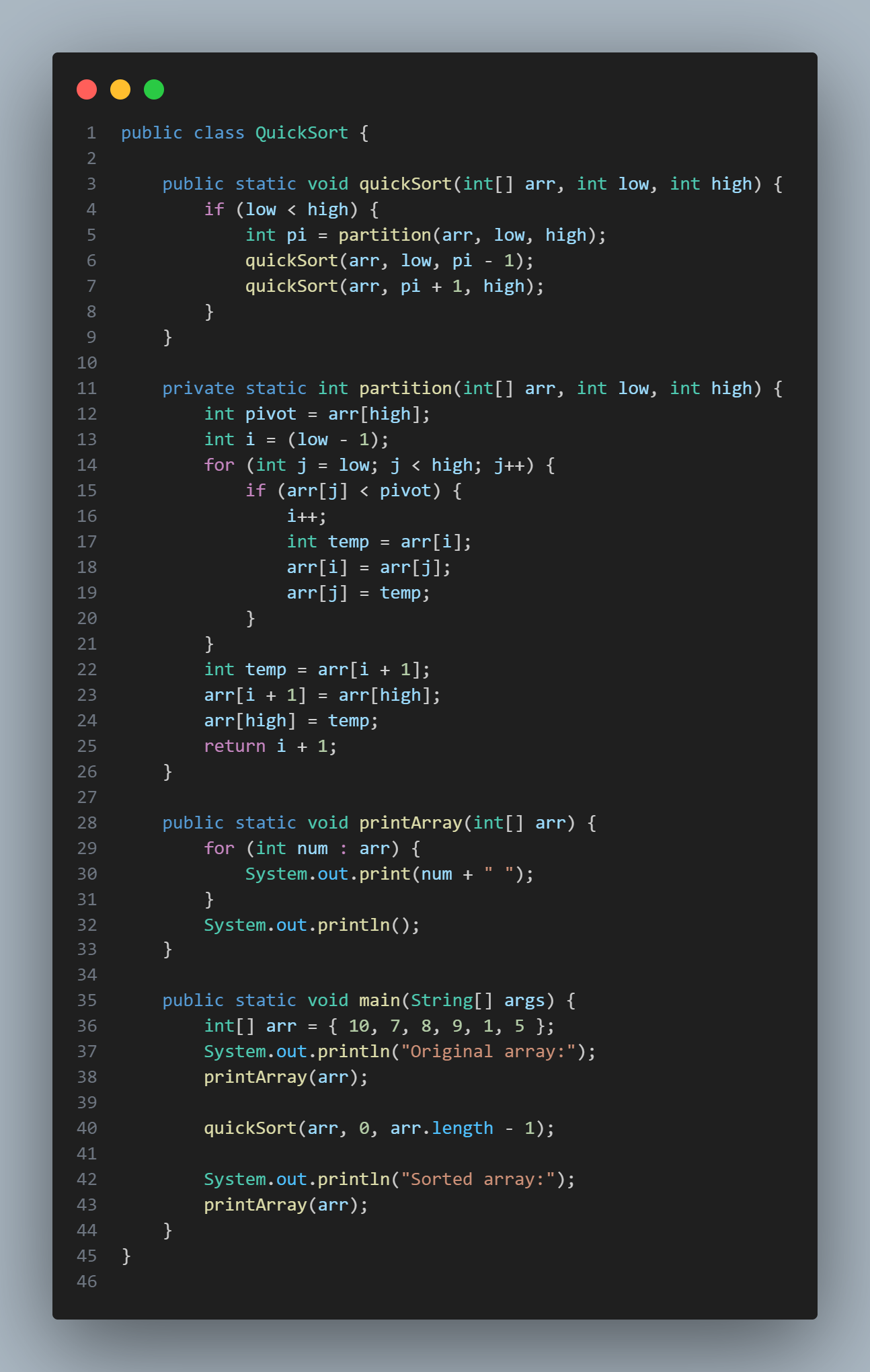
**Quicksort**

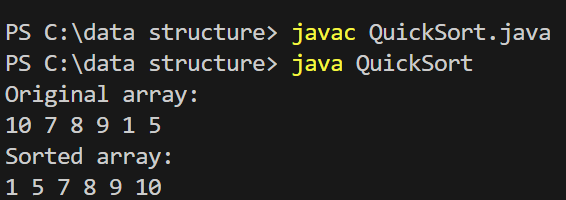
**Description: is a divide-and-conquer algorithm that picks an element as a pivot and partitions the given array around the picked pivot. The pivot element is chosen, and the array is reordered so that all elements less than the pivot come before it, and all elements greater than the pivot come after it. The same process is then applied recursively to the subarrays formed by partitioning.**

**Advantages of QuickSort:**

* **Efficiency: Average-case time complexity is O(nlog⁡n)O(n \log n), making it faster than other algorithms like Bubble Sort or Insertion Sort on large datasets.**
* **In-Place Sorting: Requires only a small amount of additional memory space (O(\log n) for the recursion stack).**
* **Versatile: Can be implemented to perform well in various scenarios by tweaking the pivot selection process.**

**Application: Suitable for large datasets due to its efficient average time complexity.**

****

****

**Bubble Sort**

**Description:** Bubble sort is one of the simplest sorting algorithms. It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

works by repeatedly iterating through the array, comparing adjacent elements and swapping them if they are in the wrong order.

**Advantages:**

* **Simplicity:** Very easy to understand and implement.
* **Stability:** Maintains the relative order of records with equal keys.
* **Adaptive:**

**Application:** Primarily for educational purposes or when the dataset is small.

