Q) What is Selection sort ? Explain with java code step by step.

→

Selection sort

Def1 :- " - It is comparison-based sorting algorithm.
- works by repeatedly finding the minimum element (for ascending order) from the unsorted part of the array and putting it at the beginning. "

Def2 :- " - smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array.

Step by Step Explanation

1) Initialization :- Start with the 1st element of the array.

2) Find minimum :- Find the minimum element in the array (or the part of the ~~elem~~ array that remains unsorted).

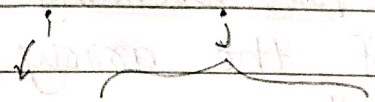
3) Swap :- Swap the found minimum element with the first element of the unsorted part of the array.

4) Repeat :- Move the boundary between sorted & unsorted parts of the array to the right, and repeat the process until the entire array is sorted.

Example :-

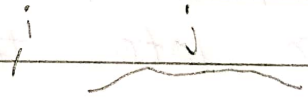
Consider an array [64, 25, 12, 22, 11]

Iteration 1



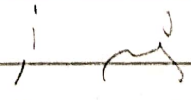
- Initial array : [64, 25, 12, 22, 11]
- Minimum element : [64, 25, 12, 22, 11] is 11.
- Swap : 11 with 64
- Array after 1st iteration : [11, 25, 12, 22, 64]

Iteration 2



- ~~Initial~~ array : [11, 25, 12, 22, 64]
- Minimum element : [25, 12, 22, 64] is 12.
- Swap : Swap 12 with 25.
- Array after 2nd iteration : [11, 12, 25, 22, 64]

Iteration 3



- ~~Int~~ Array : [11, 12, 25, 22, 64]
- Minimum element : [25, 22, 64] is 22
- Swap : 22 with 25.
- Array after 3rd iteration [11, 12, 22, 25, 64]

Iteration 4

- Array : [11, 12, 22, 25, 64]
- Minimum element : [25, 64] is 25.
- No swap needed.
- Array after 4th iteration : [11, 12, 22, 25, 64]

The array is now sorted.

Complexity

Complexity	Best Case	Average Case	Worst Case
Time	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$
Space			$O(1)$

```
public class P1_SelectionSort {
    // Function to perform selection sort.
    public static void selectionSort(int[] arr){
        int n = arr.length;
        System.out.println("Array length : " + n);

        // One by one move boundary of unsorted subarray
        for(int i=0; i<n-1; i++){
            // Find the minimum element in unsorted array
            int minIndx = i;
            for(int j=i+1; j<n; j++){
                if(arr[j] < arr[minIndx]){
                    minIndx = j;
                }
            }
            // Swap the found minimum element with the
            // 1st element
            int temp = arr[minIndx];
            arr[minIndx] = arr[i];
            arr[i] = temp;
        }
    }

    // Function to print the array.
    public static void printArray(int[] arr){
        int n = arr.length;
        for(int i=0; i<n; i++){
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
```

```
    // Main method to test the selection Sort
    public static void main(String[] args){
        int[] arr = {64, 25, 12, 22, 11};
        System.out.println("Original array: ");
        printArray(arr);

        selectionSort(arr);
        System.out.println("Sorted Array: ");
        printArray(arr);
    }
}
```

Output:-

```
Original array:
64 25 12 22 11
Array length : 5
Sorted Array:
11 12 22 25 64
```

Explanation of the Code

1. SelectionSort Method:

- The outer loop iterates over each element of the array except the last one.
- minIndex is initialized to the current index i.
- The inner loop finds the index of the minimum element in the unsorted part of the array.

- After finding the minimum element, it swaps it with the element at index i.

2. printArray Method :

- This method simply prints all elements of the array.

3. Main method :

- An example array is defined & printed.

- The selectionSort method is called to sort the array.

- The sorted array is printed.

- This is a basic implementation of the selection sort algorithm in Java.

- It sorts the array in-place, meaning it does not require any additional storage. However its time complexity is $O(n^2)$, making it inefficient for large datasets compared to more advanced algorithms like quicksort or mergesort.