

⇒ Selection Sort

↳ Pick smallest element from unsorted array,
and place it in start of unsorted array

arr =

8	6	-2	3	7
---	---	----	---	---

dry run

dry run

	i		mini		
①	8	6	-2	3	7
	-2	6	8	3	7

② -2 6 8 3 7

 i ↓ mini ↓

-2 3 8 6 7

Diagram illustrating the selection sort process:

	i	mini
3	8	6
-2	3	7
6	8	7

The diagram shows the state of the array $[-2, 3, 6, 8, 7]$ during the selection sort process. The element 3 is circled, indicating it is the current element being compared. The element 6 is marked as the "mini" (minimum) element found in the current iteration. The elements 8 and 7 are also shown, indicating the current state of the array.

④ -2 3 6 8 7

pseudo code

1) traverse 'i' from 0 to n-1

1.1) traverse 'mini'
(j) from i to n-1

1.1.1) find mini element

1.2) swap (mini, i)

code

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    selectionSort(arr, n);
}

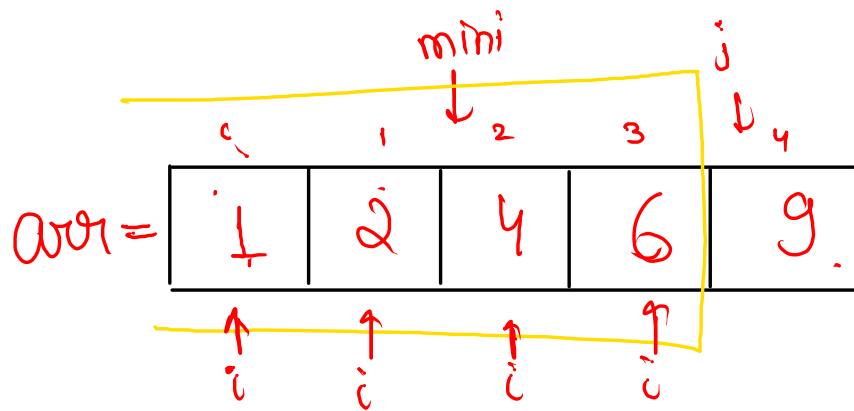
public static void selectionSort(int[] arr, int n) {
    for (int i = 0; i < n; i++) {
        int mini = i;    // index
        for (int j = i + 1; j < n; j++) {
            if (arr[j] < arr[mini]) {
                mini = j;
            }
        }
        swap(arr, i, mini);
    }

    // print
    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}

public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```

$O(N^2)$

Constant [



$i=0$

- $j=1$ ($6 < 1$) ✗
- $j=2$ ($1 < 1$) ✓✓
- $j=3$ ($9 < 1$) ✗
- $j=4$ ($2 < 1$) ✗

$i=1$

- $j=2$ ($4 < 6$) ✓✓
- $j=3$ ($9 < 4$) ✗
- $j=4$ ($2 < 4$) ✓✓

$i < n-1$

```
for (int i = 0; i < n; i++) {
    int mini = i; // index
    for (int j = i + 1; j < n; j++) {
        if (arr[j] < arr[mini]) {
            mini = j;
        }
    }
    swap(arr, i, mini);
}
```

$i=2$

- $j=3$ ($9 < 4$) ✗
- $j=4$ ($6 < 4$) ✗

$i=3$

- $j=4$ ($6 < 9$) ✓✓

HW_Kth Smallest Element

arr =

3	8	1	-2	4	0	7
---	---	---	----	---	---	---

, k = 4

ans = 3

arr =

-2	0	1	3	4	7	8
----	---	---	---	---	---	---

,

0 1 2 3

print(arr[k-1])

code

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int k = scn.nextInt();
    bubbleSort(arr, n);

    System.out.println(arr[k - 1]);
}

public static void bubbleSort(int[] arr, int n) {
    for (int i = 1; i <= n - 1; i++) {
        for (int j = 0; j < n - i; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(arr, j, j + 1);
            }
        }
    }
}

public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```

⇒ Inbuilt function

$$O(N \log(N))$$

↳ N is size of array

↳ `Arrays.sort(arr);`

↳ sort in ↑ing order

↳ `Arrays.sort(arr, Collections.reverseOrder());`

↳ sort in ↓ing order

→ Code for inbuilt function

ascending

```
public static void main(String[] args) {  
    int[] arr = { 5, 6, 3, 1, -5, 8, 2 };  
    int n = arr.length;  
  
    Arrays.sort(arr);  
  
    for (int i = 0; i < n; i++) {  
        System.out.print(arr[i] + " ");  
    }  
}
```

descending

```
public static void main(String[] args) {  
    int[] arr = { 5, 6, 3, 1, -5, 8, 2 };  
    int n = arr.length;  
    Integer[] arr1 = new Integer[n];  
    for (int i = 0; i < n; i++) {  
        arr1[i] = arr[i];  
    }  
    Arrays.sort(arr1, Collections.reverseOrder());  
    for (int i = 0; i < n; i++) {  
        System.out.print(arr1[i] + " ");  
    }  
}
```


wrapper classes
↓

int → Integer

boolean → Boolean

char → Character

double → Double

⋮

⇒ Comparator & Comparable (it will not effect T.C)

↳ used to modify the logic of inbuilt function

↳ Arrays.sort(arr)

Syntax :

code

```
public static void main(String[] args) {  
    int[] arr = { 5, 6, 3, 1, -5, 8, 2 };  
    int n = arr.length;  
    Integer[] arr1 = new Integer[n];  
    for (int i = 0; i < n; i++) {  
        arr1[i] = arr[i];  
    }  
    // comparable & comparator  
    Arrays.sort(arr1, new myComparator());  
  
    for (int i = 0; i < n; i++) {  
        System.out.print(arr1[i] + " ");  
    }  
}
```

```
public static class myComparator implements Comparator<Integer> {  
    @Override  
    public int compare(Integer a, Integer b) {  
        return a - b; // increasing order  
        // return b - a; // decreasing order  
    }  
}
```