

Sorting.

A → { 1, 5, 4, 2, 3 }

asc → .

Arrays.sort(A) → 1 2 3 4 5  
by default → asc. ✓

Arrays.sort(A);

```
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scn = new Scanner(System.in);
6         int n = scn.nextInt();
7         int [] A = new int[n];
8         for(int i = 0; i < n; i++){
9             A[i] = scn.nextInt();
10        }
11
12        //sort
13        Arrays.sort(A);
14        //print
15        for(int i = 0; i < n; i++){
16            System.out.print(A[i] + " ");
17        }
18    }
19 }
20
21
```

} i/p

} → sorting

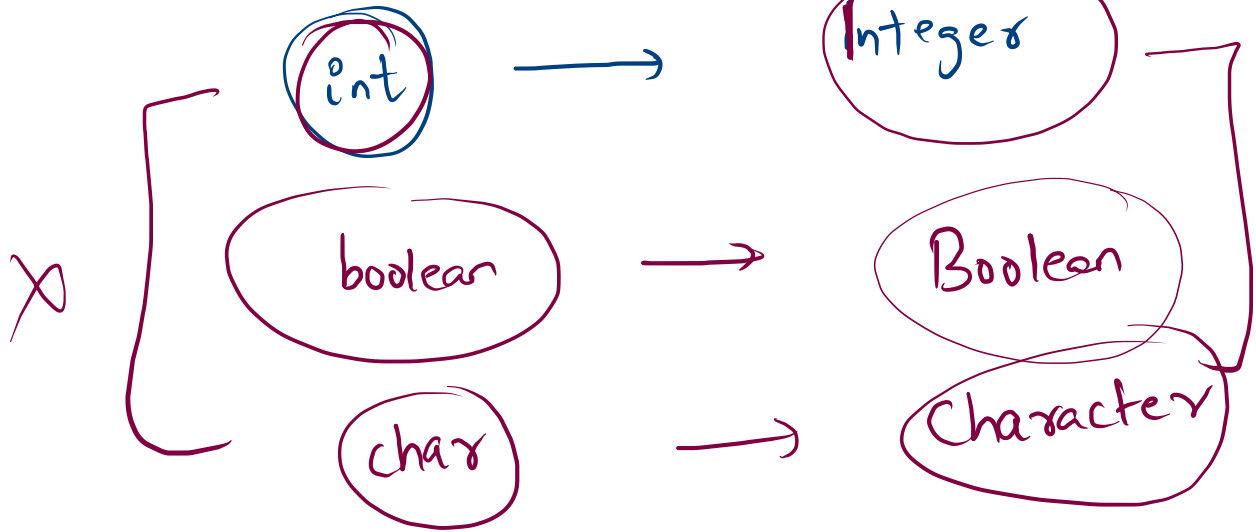
} o/p

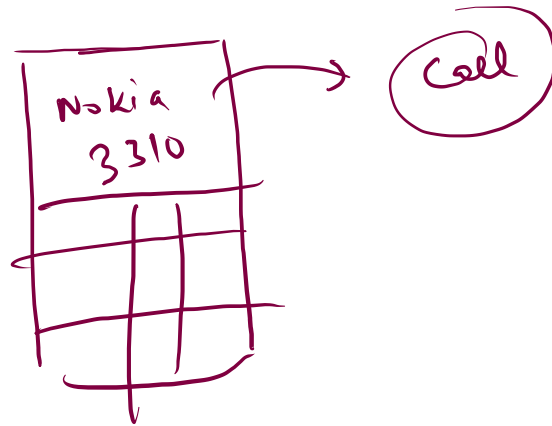
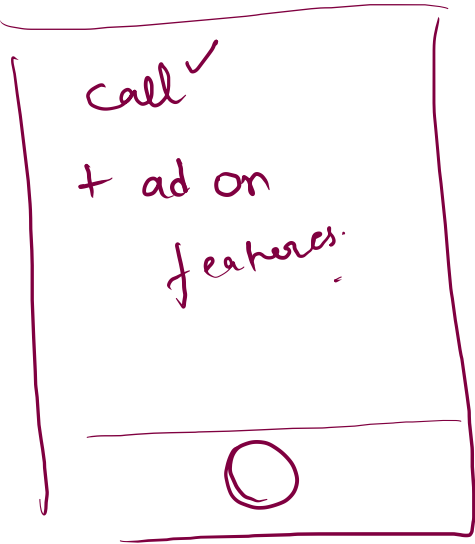
★ Collections. → reverse Order ().  
Sort → Decreasing.

issue.  
Collections. → primitive data type (X) not use  
↓  
int  
↓  
boolean  
↓  
double  
↓  
short  
↓  
long  
char.

Class / Obj.

Wrapper Class.





Integer

sort  
desc.

3	1	5	2	4
---	---	---	---	---

Collections. — primitive  
int, char, boolean

int [ ] A = new int [size];

char [ ] A = new char [size];

Integer [ ] A = new Integer [size];

## Comparators.

↳ that help us to compare

inbuilt

Collections.reverseOrder()

user defined.

my own.

```
1  import java.util.*;
2  public class Main {
3      public static void main(String [] args){
4          Scanner scn = new Scanner(System.in);
5          int n = scn.nextInt();
6          Integer [] A = new Integer[n];
7          for(int i = 0; i < n; i++){
8              A[i] = scn.nextInt();
9          }
10
11          Arrays.sort(A, Collections.reverseOrder());
12          for(int i = 0; i < n; i++){
13              System.out.print(A[i] + " ");
14          }
15      }
16  }
17  |
```

default → asc.  
desc.

Comparator.



Comparator



compare

{  
logic.  
}



compare(  
}  
  
}

Integer a,

Integer b)

int

\*\*\*

Compare

int

+ve

swap

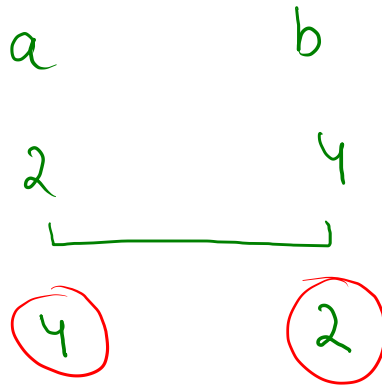
0

-ve

x  
-

asc.

desc.



int compare (a, b)

}

return 0 / -ve  
return +ve.

}

a  
4  
~~2~~

b  
2  
~~4~~

int-

Compare (a, b)

{ return.

2-4

return

4-2

}

asc. → a-b

desc → b-a

asc

$a-b$

desc

$b-a$

$a$

~~$-80$~~   
 $-14$

$b$

~~$-14$~~   
 $-80$

$-14 + 80$  ?

asc.  $\rightarrow$

desc  $\rightarrow$

$a-b$   
 $b-a$

compare  $\begin{matrix} +ve \\ 0 \\ -ve \end{matrix}$

asc.

$a-b$

desc.  
 $b-a$

$a$

30  
-23

$b$

-23  
30

```
1 import java.util.*;
2 public class Main {
3     public static void main(String [] args){
4         Scanner scn = new Scanner(System.in);
5         int n = scn.nextInt();
6         Integer [] A = new Integer[n];
7         for(int i = 0; i < n; i++){
8             A[i] = scn.nextInt();
9         }
10
11         // Syntax
12         Comparator<Integer> myComp = new Comparator<>(){
13             public int compare(Integer a, Integer b){
14                 return b-a;
15             }
16         };
17
18         Arrays.sort(A, myComp);
19         for(int i = 0; i < n; i++){
20             System.out.print(A[i] + " ");
21         }
22     }
23 }
24
```

Sort by square of element.

asc.  
a-b

4	-1	0	-5	6
16	①	0	25	36
0	1	2	3	4

### Test Case 1

Input:

```
5
4 -1 0 -5 6
```

Output:

```
0 -1 4 -5 6
```

0 -1 4 -5 6

?

```
1  import java.util.*;
2  public class Main {
3      public static void main(String [] args){
4          Scanner scn = new Scanner(System.in);
5          int n = scn.nextInt();
6          Integer [] A = new Integer[n];
7          for(int i = 0; i < n; i++){
8              A[i] = scn.nextInt();
9          }
10
11         // Syntax
12         Comparator<Integer> myComp = new Comparator<>(){
13             public int compare(Integer a, Integer b){
14                 return a*a - b*b;
15             }
16         };
17
18         Arrays.sort(A, myComp);
19         for(int i = 0; i < n; i++){
20             System.out.print(A[i] + " ");
21         }
22     }
23 }
```



A →

2	1	7	5	3	4	8
0	1	2	3	4	5	6

A green bracket is drawn under the elements at indices 1 through 4 (values 1, 7, 5, 3).

sort (A, 3, 7) → 2 1 7    3 4 5 8 ✓

\* sort (A, 1, 5) → 2    1 3 5 7    4 8

# Sort Array By Parity

Sort Array By Parity

## Problem Statement

Given an integer array `nums[]`, move all the even integers at the beginning of the array followed by all the odd integers in non- decreasing order.

nums ->

3	1	2	4
o	o	e	e

## Test Case 1

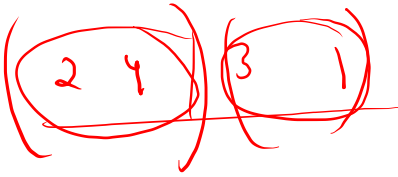
Input:

```
4
3 1 2 4
```

Output:

```
2 4 1 3
```

3	2	1	4
---	---	---	---



2 4      1 3

(all even)... (all odd)

3	2	4	1
---	---	---	---

★

1. segregate even... odd.
2. sort even range ✓
3. sort odd range ✓

2 4    3 1

2 4    3 1

2 4 1 3