Sorting.

A → { 1 , 5 , 4 , 2 , 3 }

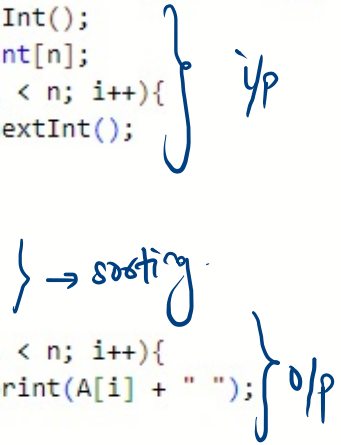
asc →

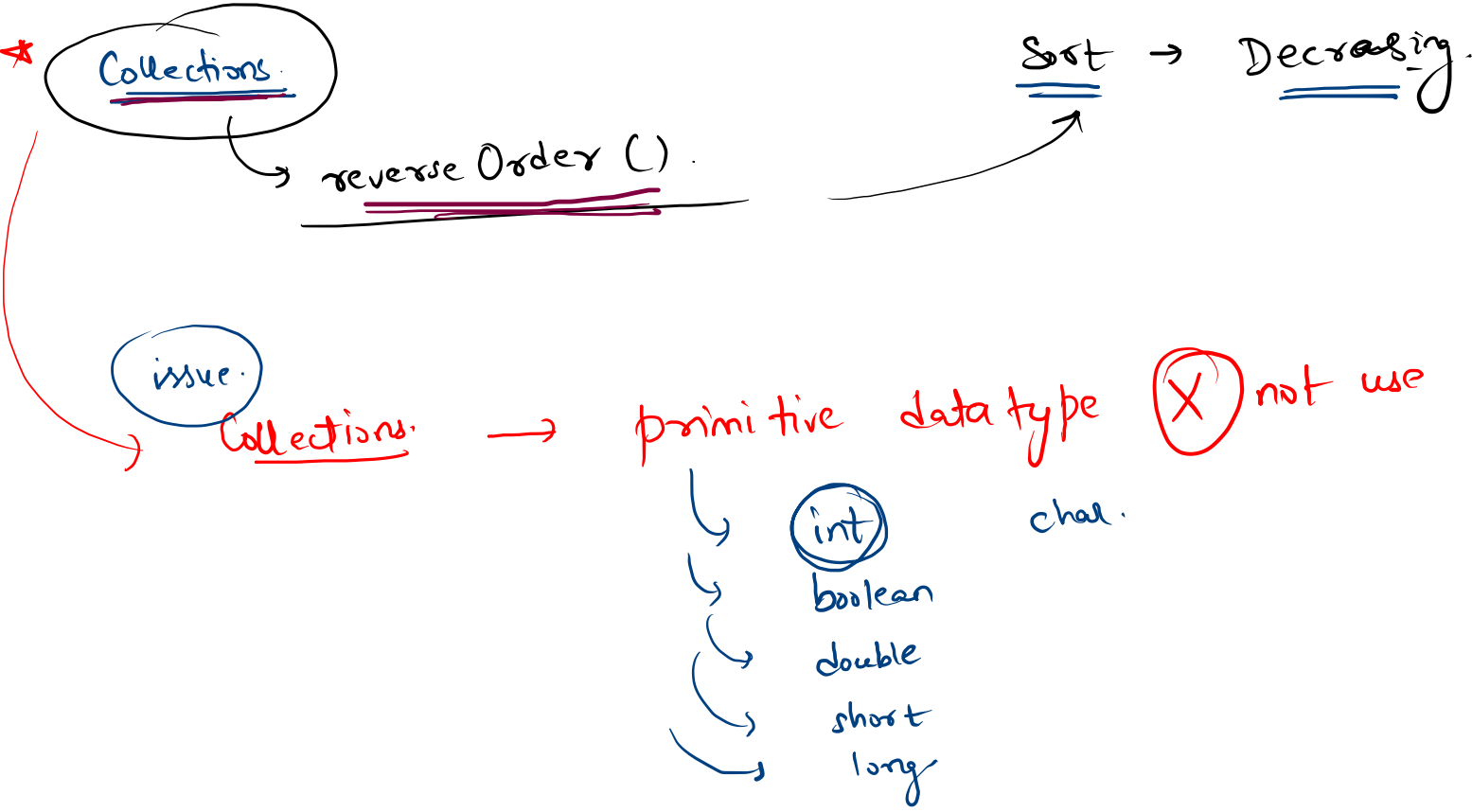
Arrays.sort (A) → 1 2 3 4 5

by default → asc. ✓

```
Arrays.sort (A);
```

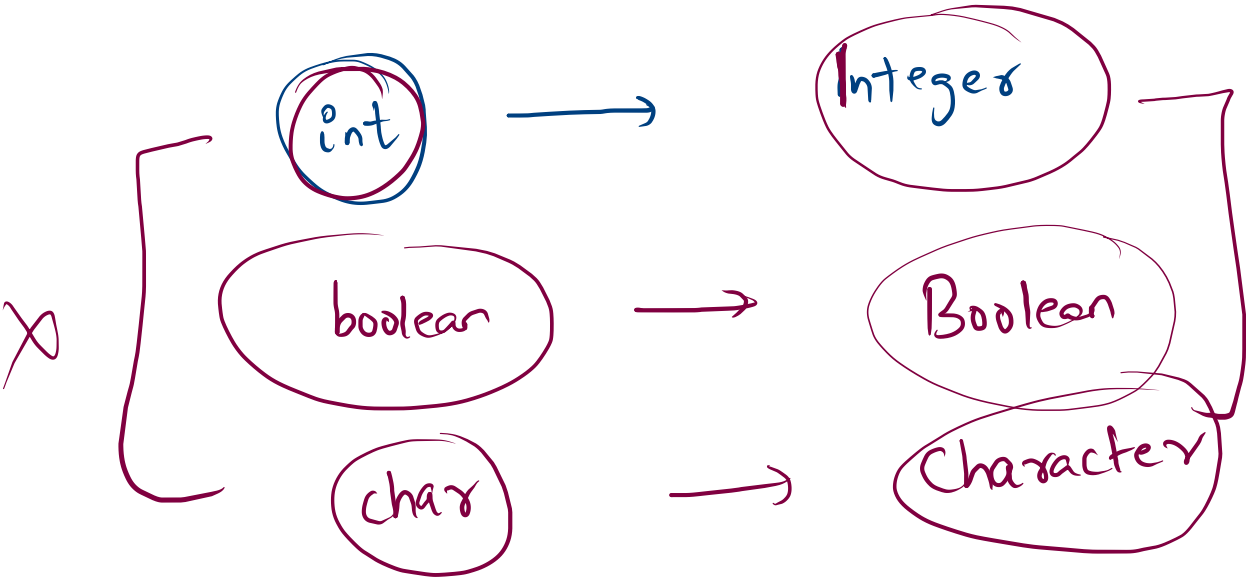
```java
public class Main {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }

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

    }
}
```

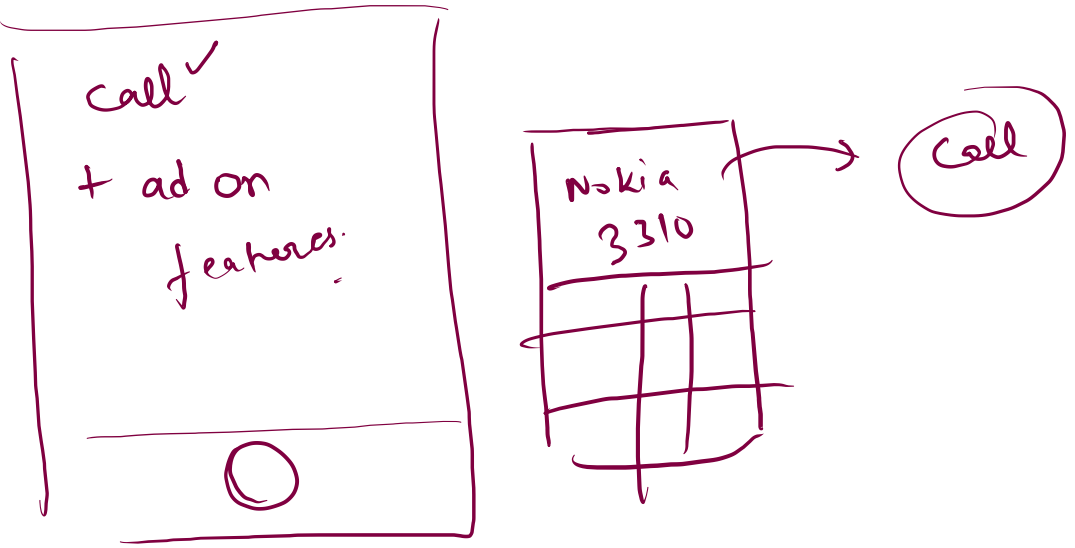



Collections.
reverse Order ().
Sort → Decreasing.
issue.
Collections. → primitive data type (X) not use
int
boolean
double
short
long
char.

Class / Obj.

Wrapper Class.




call ✓
+ ad on
features.
Nokia
3310
call

Sort

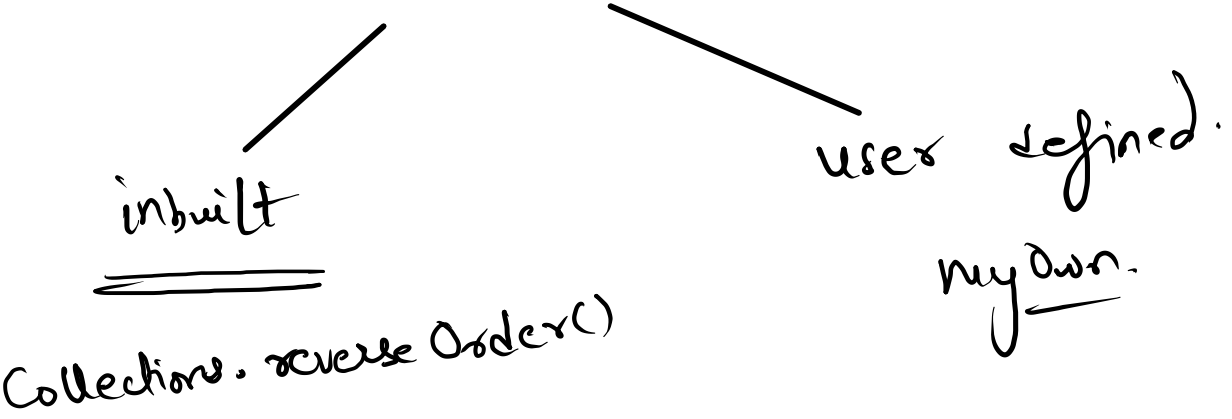
desc.

Collections. — primitive
\ int, char, boolean

```
int [ ]    A  =  new  int [ size];
char [ ]   A  =  new  char [ size];
Integer [ ]  A =  new  Integer [ size];
```

Comparator.

↳ that help us to compare



default → asc.
desc.

```java
import java.util.*;
public class Main {
    public static void main(String [] args){
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        Integer [] A = new Integer[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }

        Arrays.sort(A, Collections.reverseOrder());
        for(int i = 0; i < n; i++){
            System.out.print(A[i] + " ");
        }
    }
}
```

Comparator.

Comparator

Compare

{

logic.

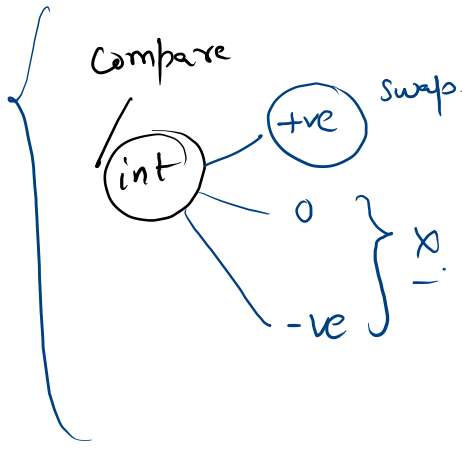
}

int compare(Integer a, Integer b)

{

}

***



a b

2 4

asc.

desc.

int compare (a, b)

{

return 0/-ve

return +ve.

}

a b

4 2

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. → a - b

desc → b - a

compare +ve 0 -ve

a

30

-23

b

-23

30

asc.

a - b.

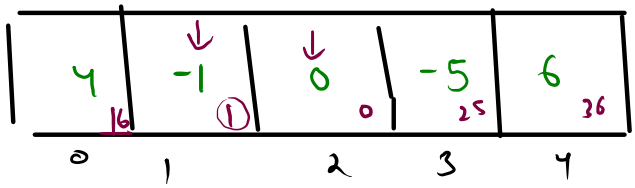
desc.

b - a.

```java
import java.util.*;
public class Main {
    public static void main(String [] args){
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        Integer [] A = new Integer[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }

        // Syntax
        Comparator<Integer> myComp = new Comparator<>(){
            public int compare(Integer a, Integer b){
                return b-a;
            }
        };

        Arrays.sort(A, myComp);
        for(int i = 0; i < n; i++){
            System.out.print(A[i] + " ");
        }
    }
}

```

Sort by square of element.

asc.
↳ a-b



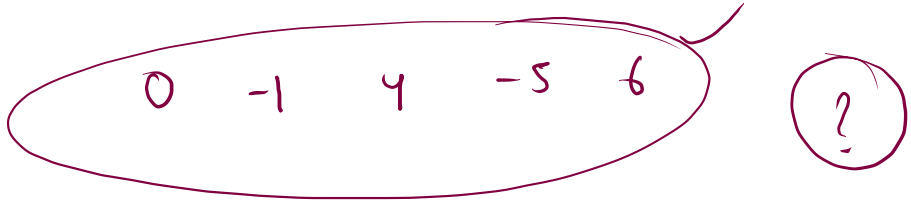
Test Case 1

Input:

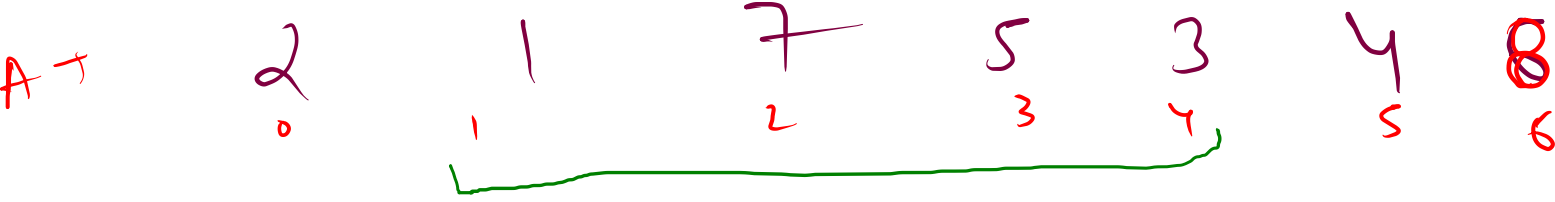
```
5
4 -1 0 -5 6
```

Output:

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



```java
import java.util.*;
public class Main {
    public static void main(String [] args){
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        Integer [] A = new Integer[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }

        // Syntax
        Comparator<Integer> myComp = new Comparator<>(){
            public int compare(Integer a, Integer b){
                return a*a - b*b;
            }
        };

        Arrays.sort(A, myComp);
        for(int i = 0; i < n; i++){
            System.out.print(A[i] + " ");
        }
    }
}
```



sort (A, 3, 7) → 217 3458 ✓

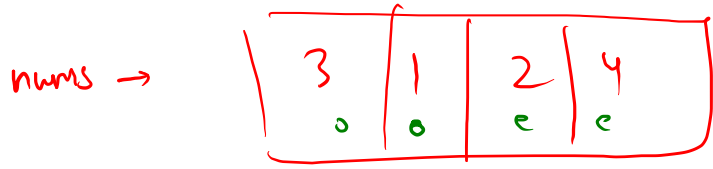
* sort (A, 1, 5) → 2 1 3 5 7 4 8

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.



(all even)... (all odd)

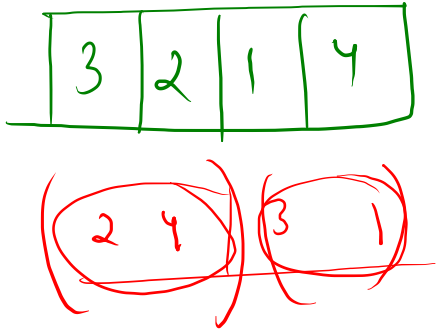
Test Case 1

Input:

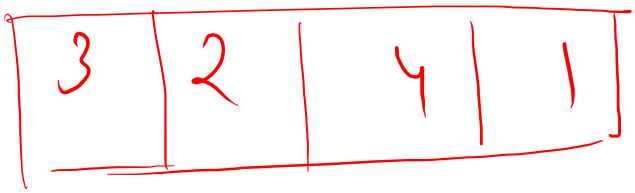
```
4
3 1 2 4
```

Output:

```
2 4 1 3
```



2 4 1 3



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

2 4 3 1

2 4 3 1

2 4 1 3