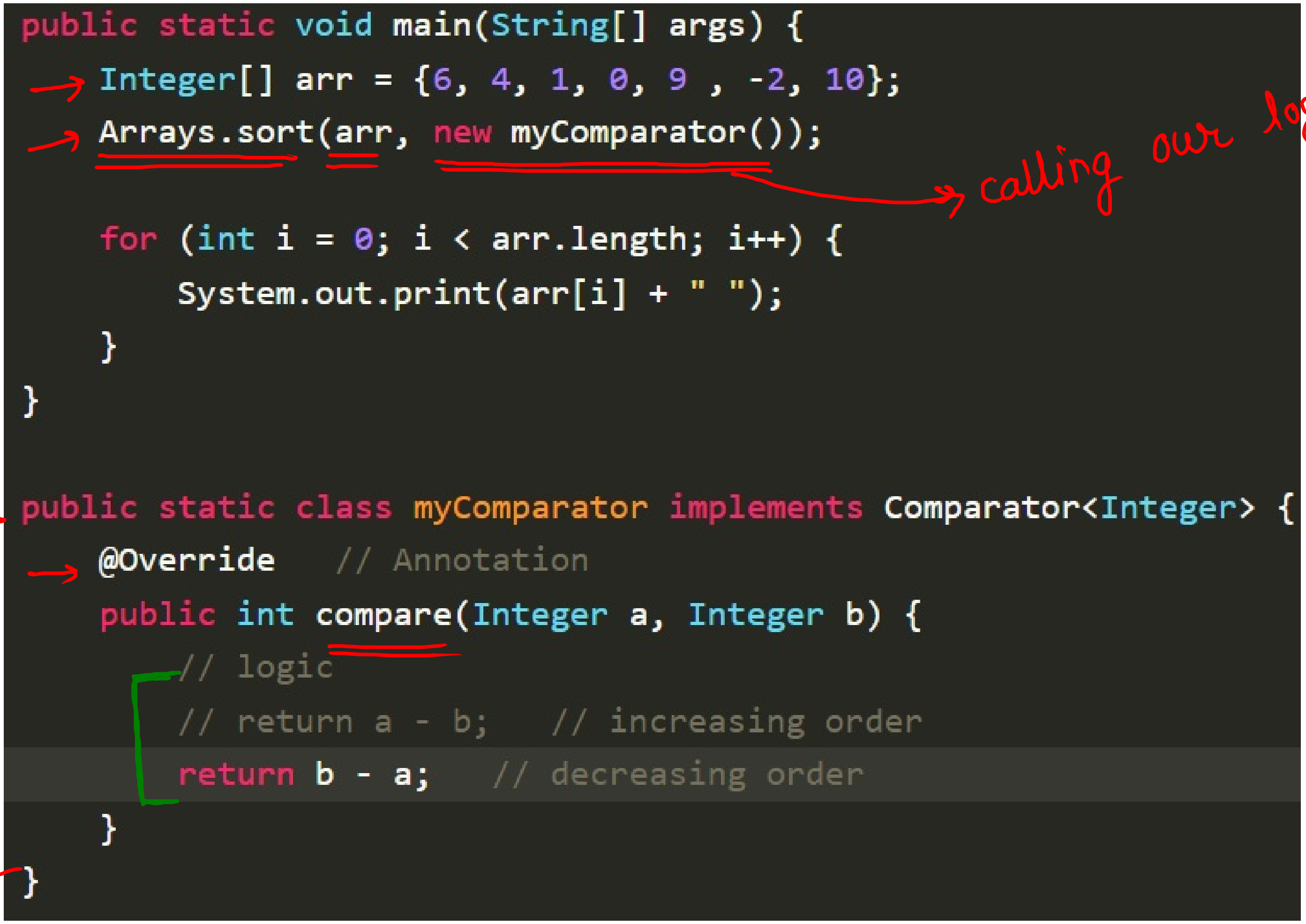
⇒ **Comparator and Comparable** (Inbuilt Sort)

↳ here, we can modify the logic of Inbuilt function

```
Arrays.sort(arr);

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



```java
public static void main(String[] args) {
    Integer[] arr = {6, 4, 1, 0, 9 , -2, 10};
    Arrays.sort(arr, new myComparator());

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

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

calling our logic

Implimenting our logic

Trick :-

a → myself
b → other

$a - b = (-1)$
5 10 increasing

$b - a = (+1)$
10 5 decreasing

Keyword :- new

('new' is used to create an object)

object is anything

Sort the array according to their Square of each element

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

    // main logic
    Arrays.sort(arr, new myComparator());

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

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

$\underbrace{a * a}_{a} - \underbrace{b * b}_{b}$ $\longrightarrow$ main logic

n = 5

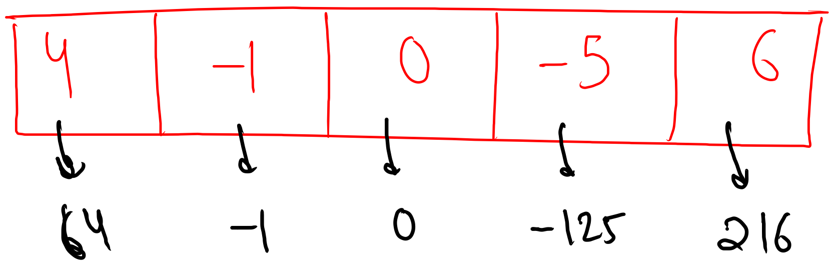
4	-1	0	-5	6
↓	↓	↓	↓	↓
16	1	0	25	36

in ascending order acc. to sq. values

after sorting

0	-1	4	-5	6

Ques 5



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

    // main logic
    Arrays.sort(arr, new myComparator());

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

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

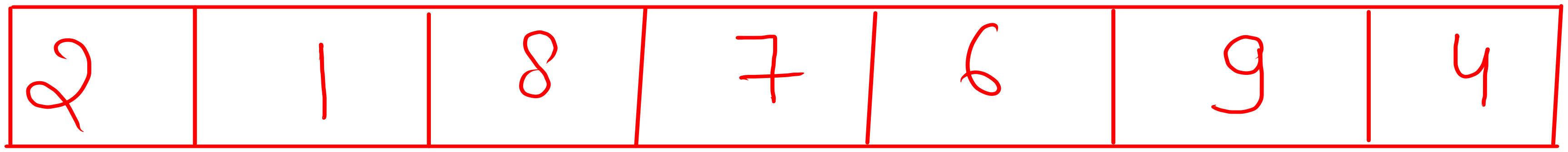
ans:-

-5	-1	0	4	6

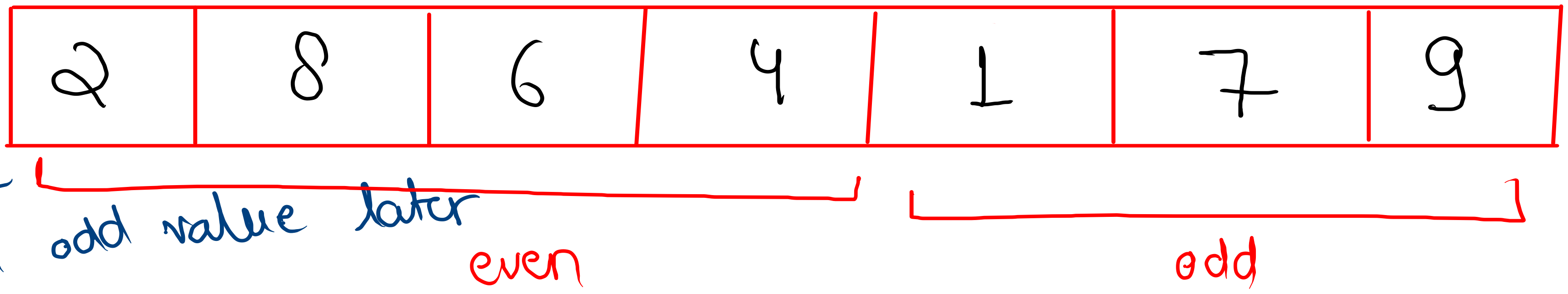
after sorting

Sort Array By Parity

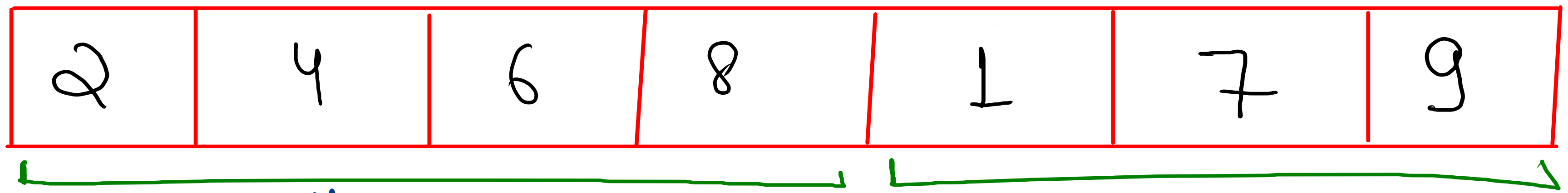
arr



arrange array like even value first and odd value later



arrange array in non-dec. order as well



2	1	8	7	6	9	4

Conditions:- (consider any 2 values to compare)

a = even	b = even
a = odd	b = odd
a = even	b = odd
a = odd	b = even

1) a = even, b = odd (−1)

2) a = even, b = even smaller value should be appear first (a − b)

3) a = odd, b = odd smaller value should be appear first (a − b)

4) a = odd, b = even (+1)

Code

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

    // main logic
    Arrays.sort(arr, new myComparator());

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

public static class myComparator implements Comparator<Integer> {
    @Override
    public int compare(Integer a, Integer b) {

        if ( a % 2 == 0 && b % 2 != 0 ) {   //a = even, b = odd
            return -1;
        } else if (a % 2 != 0 && b % 2 == 0) {  // a = odd, b = even
            return 1;
        } else if (a % 2 == 0 && b % 2 == 0) {  // a = even, b = even
            return a - b;
        } else {  // a = odd, b = odd
            return a - b;
        }

    }
}
```

⇒ Lambda function

```
Arrays.sort(arr, (a, b) -> {
        return a-b;   // inc.
});
```

Sort the array according to their Square of each element

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

    // main logic
    Arrays.sort(arr, (a, b) -> {
        return a * a - b * b;
    });

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

Sort Array By Parity

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

    // main logic
    Arrays.sort(arr, (a, b) -> {
        if ( a % 2 == 0 && b % 2 != 0 ) {   //a = even, b = odd
            return -1;
        } else if (a % 2 != 0 && b % 2 == 0) {  // a = odd, b = even
            return 1;
        } else if (a % 2 == 0 && b % 2 == 0) {  // a = even, b = even
            return a - b;
        } else {  // a = odd, b = odd
            return a - b;
        }
    });

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

1 5 6 7 10 9 8

arr

sort

ans 1 5 7 9 10 8 6

$$\begin{tikzcd}
\text{odd} \arrow[r] \arrow[d] & \text{even} \arrow[d] \\
\text{non-dec} & \text{non-inc}
\end{tikzcd}$$

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

    // main logic
    Arrays.sort(arr, (a, b) -> {
        if ( a % 2 == 0 && b % 2 != 0 ) {    //a = even, b = odd
            return ~~-1~~;  +1;
        } else if (a % 2 != 0 && b % 2 == 0) {  // a = odd, b = even
            return ~~1~~;  -1;
        } else if (a % 2 == 0 && b % 2 == 0) {  // a = even, b = even
            return ~~a - b~~;   b-a
        } else {  // a = odd, b = odd
            return a - b;
        }
    });

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