??(Sort)

- Comparator + 람다식 → 커스텀 정렬
- Map 정렬
- PriorityQueue 정렬 기준
- Java Stream 정렬

Comparator + ??? ? ??? ??

---

1. ?? ??

```
Arrays.sort(배열, (a, b) -> {
    return 비교값;
});
```

- 자바에서 배열을 정렬할 때, `Comparator`를 람다식으로 넘겨 정렬 기준을 직접 지정할 수 있다.
- Comparator는 객체 타입인 Integer에만 사용할 수 있기 때문에 int[] → Integer[] 변환이 필요할 때가 있다.

1.1 Comparator ??? ??? ??

반환값	의미	정렬 결과
음수 (`< 0`)	a가 b보다 작다	a가 앞에 옴
0	a와 b가 같다	순서 유지
양수 (`> 0`)	a가 b보다 크다	b가 앞에 옴

---

2. ?? ???? ??? ??

2.1 ???? ?? (?? ? ? ? ?)

```
Arrays.sort(arr, (a, b) -> a - b);
```

2.2 ???? ?? (? ? ? ?? ?)

```
Arrays.sort(arr, (a, b) -> b - a);
```

2.3 ??? ?? ???? ??

```
Arrays.sort(arr, (a, b) -> Math.abs(a) - Math.abs(b));
```

2.4 ??? ?? + ?? ??? ? ? ??

```
Arrays.sort(arr, (a, b) -> {
    int diff = Math.abs(a) - Math.abs(b);
    if (diff == 0) return b - a; // 절댓값 같으면 큰 수 우선
    return diff;
});
```

---

3. ??

목적	정렬식
기본 오름차순	`(a, b) -> a - b`
기본 내림차순	`(a, b) -> b - a`
절댓값 기준	`(a, b) -> Math.abs(a) - Math.abs(b)`
절댓값 + 값이 같으면 큰 수 우선	`(a, b) -> { ... if문으로 처리 }`

---

4. List ??

4.1 ?? ???? ??

```
List<Integer> list = Arrays.asList(3, 1, 5, 2);
Collections.sort(list);  // 기본 오름차순
```

4.2 ???? ?? (??? ??)

```
list.sort((a, b) -> b - a);  // 또는 Collections.sort(list, (a, b) -> b - a);
```

4.3 ??? ?? ??

```
list.sort((a, b) -> Math.abs(a) - Math.abs(b));
```

---

5. ?? ??

5.1 ?? 1: Comparable ????? (?? ?? ?? ??)

?? ???

```
class Person implements Comparable<Person> {
    String name;
    int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public int compareTo(Person other) {
        return this.age - other.age;  // □□ □□□□ □□
    }
}
```

?? ??

```
List<Person> list = new ArrayList<>();
list.add(new Person("Alice", 25));
list.add(new Person("Bob", 20));
Collections.sort(list);  // compareTo() □□□□ □□□
```

5.2 ?? 2: Comparator ????? (?? ?? ??)

?? 1: ?? ?? ?? (????)

```
list.sort(Comparator.comparing(p -> p.name));
```

?? 2: ?? ???? ??

```
list.sort((p1, p2) -> p2.age - p1.age);
```

---

6. ?? ??

구분	인터페이스	사용 위치	정렬 기준 변경
Comparable	`compareTo()`	클래스 내부 정의	고정 (자기 기준)
Comparator	`compare()`	외부에서 전달	유동적 (다양한 기준 가능)

- `Comparable`: "나는 정렬 방법을 내가 정한다!" (클래스 안에서 정의)
- `Comparator`: "정렬 기준은 밖에서 주입한다!" (상황에 따라 유연하게)

Map ??

1. Map? ? ???? ??

```
Map<String, Integer> map = new HashMap<>();
map.put("apple", 3);
map.put("banana", 1);
map.put("cherry", 2);

// 키 기준 오름차순 정렬
Map<String, Integer> sortedByKey = new TreeMap<>(map);
```

2. Map? ? ???? ??

```
List<Map.Entry<String, Integer>> entries = new ArrayList<>(map.entrySet());

// 값 기준 오름차순 정렬
entries.sort(Map.Entry.comparingByValue());

// 값 기준 내림차순 정렬
entries.sort((a, b) -> b.getValue() - a.getValue());
```

3. ??? Map ???

```
Map<String, Integer> sortedMap = new LinkedHashMap<>();
for (Map.Entry<String, Integer> entry : entries) {
    sortedMap.put(entry.getKey(), entry.getValue());
}
```

PriorityQueue ?? ??

1. ???? ???? ? (?? ?? ??)

```
PriorityQueue<Integer> pq = new PriorityQueue<>();
pq.add(5);
pq.add(2);
pq.add(8);
```

2. ???? ???? ? (? ?? ??)

```
PriorityQueue<Integer> pq = new PriorityQueue<>((a, b) -> b - a);
```

3. ??? ?? ?? ??

```
class Person {
    String name;
    int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}

PriorityQueue<Person> pq = new PriorityQueue<>((p1, p2) -> p1.age - p2.age); // □□ □□□□
```

Java Stream ??

1. ?? ?? (?? ????)

```
List<Integer> list = Arrays.asList(5, 3, 1, 4);
List<Integer> sorted = list.stream()
    .sorted()
    .collect(Collectors.toList());
```

2. ???? ??

```
List<Integer> sortedDesc = list.stream()
    .sorted(Comparator.reverseOrder())
    .collect(Collectors.toList());
```

3. ?? ??? ??

```
List<Person> people = Arrays.asList(
    new Person("Alice", 25),
    new Person("Bob", 20)
);

// 나이 기준 오름차순
List<Person> sortedPeople = people.stream()
    .sorted(Comparator.comparing(p -> p.age))
    .collect(Collectors.toList());
```

4. ?? ???? ??

```
// 나이 기준, 같으면 이름 기준
.sorted(Comparator.comparing(Person::getAge)
                  .thenComparing(Person::getName))
```