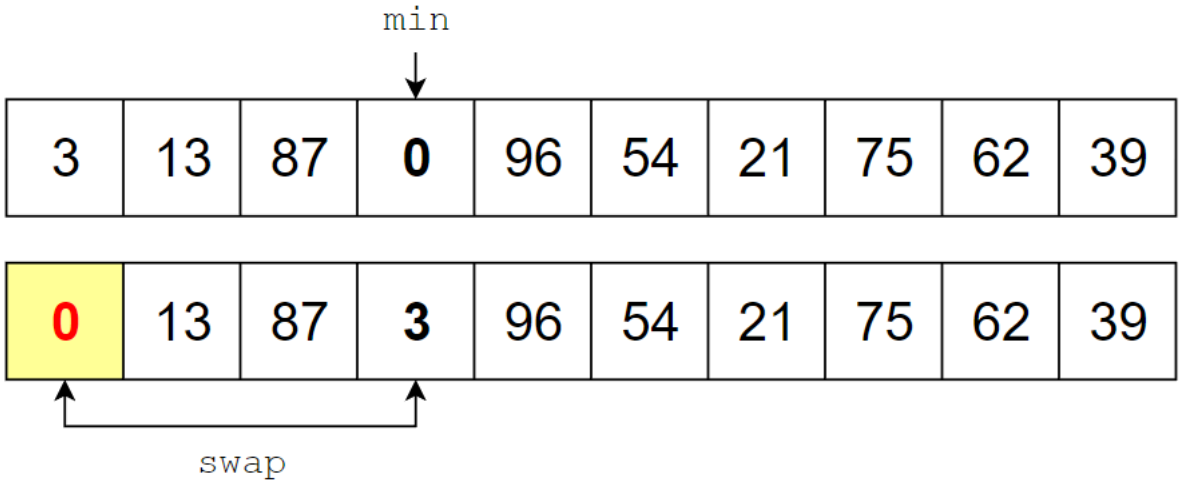
Sorting(정렬)

- 검색(Search)이 빠르다.....



1. 배열 정렬 (Arrays.sort())

```
import java.util.Arrays;

public class SortExample {
    public static void main(String[] args) {
        int[] numbers = {5, 2, 8, 1, 9};
        Arrays.sort(numbers); // 오름차순 정렬
        System.out.println(Arrays.toString(numbers)); // 출력: [1, 2, 5, 8, 9]
    }
}
```

- 내림차순

```
import java.util.Arrays;
import java.util.Collections;

public class SortExample {
    public static void main(String[] args) {
        Integer[] numbers = {5, 2, 8, 1, 9};
        // numbers 는 참조 타입
```

```
        Arrays.sort(numbers, Collections.reverseOrder()); // 내림차순 정렬
        System.out.println(Arrays.toString(numbers)); // 출력: [9, 8, 5, 2, 1]
    }
}
```

- 정렬 범위 지정

```
import java.util.Arrays;

public class SortExample {
    public static void main(String[] args) {
        int[] numbers = {5, 2, 8, 1, 9, 4, 7};
        Arrays.sort(numbers, 1, 5); // 인덱스 1부터 4까지 정렬 (5는 미포함)
        System.out.println(Arrays.toString(numbers)); // 출력: [5, 1, 2, 8, 9, 4, 
    }
}
```

2. 리스트 정렬 (Collections.sort())

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class SortExample {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>(Arrays.asList(5, 2, 8, 1, 9));
        Collections.sort(numbers); // 오름차순 정렬
        System.out.println(numbers); // 출력: [1, 2, 5, 8, 9]
    }
}
```

- 내림차순 정렬

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class SortExample {
```

```
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>(Arrays.asList(5, 2, 8, 1, 9));
        Collections.sort(numbers, Collections.reverseOrder()); // 내림차순 정렬
        System.out.println(numbers); // 출력: [9, 8, 5, 2, 1]
    }
}
```

3. 객체 비교 -Comparable Interface

- comparable

```
import java.util.Arrays;

class Rectangle implements Comparable<Rectangle> {
    private int width, height;

    public Rectangle(int width, int height) {
        this.width = width;
        this.height = height;
    }

    public int findArea() {
        return width * height;
    }

    public String toString() {
        return String.format("사각형[폭=%d, 높이=%d]", width, height);
    }
    @Override
    public int compareTo(Rectangle o) {
        return findArea() - o.findArea();
    }
}

public class ComparableDemo {
    public static void main(String[] args) {
        Rectangle[] rectangles = { new Rectangle(3, 5),
```

```
            new Rectangle(2, 10), new Rectangle(5, 5) };

        Arrays.sort(rectangles);

        for (Rectangle r : rectangles)
            System.out.println(r);
    }
}
```

```
public class Circle implements Comparable<Circle>{
    private double radius;
    final double PI=3.14;

    public Circle(double radius) {
        this.radius = radius;
    }
    public double findArea() {
        return radius * radius *PI;
    }
    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return String.format("원[반지름=%f] 면적=%f"
                , radius, findArea());
    }
    @Override
    public int compareTo(Circle o) {
        // TODO Auto-generated method stub
        return (int)(findArea() - o.findArea());
    }
}
```

- Comparator
  - Comparator는 사용자 정의 객체를 사용하거나 클래스 내에서 비교 메서드를 구현하고자 할 때
  - Comparator 인터페이스를 상속받게되면, 반드시 compare함수를 Override 해줘야 합니다

```
public interface Comparator<T>{
   int compare(T o1, T o2);
}
```

```
import java.util.Arrays;
import java.util.Comparator;

public class ComparatorDemo {
   public static void main(String[] args) {
      String[] strings = { "로마에 가면 로마법을 따르라.",
            "시간은 금이다.", "펜은 칼보다 강하다." };

      Arrays.sort(strings, new Comparator<String>() {
         public int compare(String first, String second) {
            return first.length() - second.length();
         }
      });

      for (String s : strings)
         System.out.println(s);
   }
}
```

학생 점수 정렬

```
100, 홍길동,90, 77,88
101, 이순신,88,94, 90
102,타이거, 78,88,99
103,라이온,85,90,100
```

```
public class Student (_____){
   private String hakbun;
   private String name;
   private int korean, eng, math;
```

```
    public Student(String hakbun, String name, int korean, int eng, int math) {
        this.hakbun = hakbun;
        this.name = name;
        this.korean = korean;
        this.eng = eng;
        this.math = math;
    }
    public int getTotal(){
        return korean + eng +math;
    }
    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return String.format("%s %s %d ",  hakbun, name, getTotal());;
    }
    @Override
    public int compareTo(Student o) {
        return (_____________________________);
    }
}
```