
Arrays 클래스

Arrays 클래스

- 배열의 복사
  - 배열의 복사를 위해 Arrays 클래스에 정의되어 있는 메소드
    - public static int[ ] copyOf(int [ ] original, int newLength)
    - original에 전달된 배열을 첫 번째 요소부터 newLength의 길이만큼 복사

```
public class CopyOfArrays {
    public static void main(String[] args) {
        double[] arOrg = {1.1, 2.2, 3.3, 4.4, 5.5};
        double[] arCpy1 = Arrays.copyOf(arOrg, arOrg.length);   // 배열 전체 복사
        double[] arCpy2 = Arrays.copyOf(arOrg, 3);              // 첫 번째 요소에서 세 번째 요소까지 복사
        for(double d : arCpy1) {
            System.out.print(d + "₩t");
        }
        System.out.println();
        for(double d : arCpy2) {
            System.out.println(d + "₩t");
        }
    }
```

Arrays 클래스

- 배열의 일부 복사를 위해 Arrays 클래스에 정의되어 있는 메소드
  - public static int[ ] copyRange(int[ ] original, int from, int to)
  - 파라메터 to에 명시된 정수의 이전 요소까지 복사

```
public class RangeCopyOfArrays {
    public static void main(String[] args) {
        double[] arOrg = {1.1, 2.2, 3.3, 4.4, 5.5};
        double[] arCpy = Arrays.copyOfRange(arOrg, 1, 3);   // 배열 전체 복사
        for(double d : arCpy) {
            System.out.print(d + "₩t");
        }
    }
}
```

Arrays 클래스

- 배열의 비교
  - 배열의 비교를 위해 Arrays 클래스에 정의되어 있는 메소드
    - public static boolean equals(int[ ] a, int[ ] b)
    - 파라메터 a와 b로 전달된 배열의 내용을 비교하여 true, false 반환
    - 두 배열에 저장된 데이터의 수, 순서, 내용이 같을 때 true 반환
    - 클래스의 인스턴스 배열 비교 가능

```
public class ArrayEquals {
    public static void main(String[] args) {
        int[] ar1 = {1, 2, 3, 4, 5};
        int[] ar2 = {1, 2, 3, 4};
        int[] ar3 = {1, 2, 3, 4, 5};
        System.out.println(Arrays.equals(ar1, ar2));
        System.out.println(Arrays.equals(ar1, ar3));
    }
}
```

Arrays 클래스

- 배열의 정렬
  - 배열의 정렬을 위해 Arrays 클래스에 정의되어 있는 메소드
    - public static boolean sort(int[ ] a)
    - 파라메터 a로 전달된 배열을 오름차순으로 정렬

```
public class ArraySort {
    public static void main(String[] args) {
        int[] ar1 = {1, 5, 3, 2, 4};
        double[] ar2 = {3.3, 2.2, 5.5, 1.1, 4.4};
        Arrays.sort(ar1);
        Arrays.sort(ar2);
        for(int n : ar1) {
            System.out.println(n + "₩t");
        }
        System.out.println();
        for(double d : ar2) {
            System.out.println(d + "₩t");
        }
    }
}
```

Arrays 클래스

- 클래스의 인스턴스 배열에 대해서도 정렬 가능
- Comparable 인터페이스의 구현을 통해 인스턴스 배열의 정렬을 위한 기준 구현 필요
- Comparable 인터페이스의 추상 메소드 구현 : int compareTo(Object o)
  - 인자로 전달된 o가 작다면 양의 정수 반환
  - 인자로 전달된 o가 크다면 음의 정수 반환

```
public class Person implements Comparable {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    @Override
    public int compareTo(Object o) {
        Person p = (Person) o;
        if (this.age > p.age) {return 1};
        else if (this.age < p.age) {return -1;}
        else {return 0;}
    }
    @Override
    public String toString() {
        return this.name + " : " + this.age;
    }
}
```

```
public class ArrayObjSort {
    public static void main(String[] args) {
        Person[] ar = new Person[3];
        ar[0] = new Person("JungMinWook", 29);
        ar[1] = new Person("LeeSangSun", 40);
        ar[2] = new Person("Gariyong", 15);
        Arrays.sort(ar);
        for(Person p : ar) {
            System.out.println(p);
        }
    }
}
```

Arrays 클래스

1. 앞의 예제에서 Person 클래스의 인스턴스들을 나이의 올림차순으로 정렬하였는데, 이를 나이의 내림차순으로 정렬하도록 예제를 수정해보자.
2. 앞의 예제에서 Person 클래스의 인스턴스들을 나이의 올림차순으로 정렬하였는데, 이를 이름의 길이 순으로 정렬하도록 예제를 수정해보자.

Arrays 클래스

- 배열의 탐색
  - 배열의 탐색를 위해 Arrays 클래스에 정의되어 있는 메소드
    - public static int binarySearch(int[ ] a, int key)
    - 배열 a에서 key를 찾아서 있으면 key의 인덱스 값, 없으면 0보다 작은 수 반환
    - 이진 탐색 알고리즘 기반으로 탐색을 진행하므로 오름차순 정렬 진행 후 탐색

```
public class ArraySearch {
    public static void main(String[] args) {
        int[] ar = {3, 5, 1, 4, 2};
        Arrays.sort(ar);
        for(int n : ar) {
            System.out.print(n + "₩t");
        }

        int inx = Arrays.binarySearch(ar, 3);
        System.out.println("Index of 33 : " + idx);
    }
}
```

Arrays 클래스

- 클래스의 인스턴스 배열에 대해서도 탐색 가능
  - public static int binarySearch(Object[] a, Object key) 오버로딩
- Comparable 인터페이스의 구현을 통해 인스턴스 배열의 탐색을 위한 기준 구현 필요
- Comparable 인터페이스의 추상 메소드 구현 : int compareTo(Object o)

```
public class Person implements Comparable {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    @Override
    public int compareTo(Object o) {
        Person p = (Person) o;
        return this.age - p.age; // 나이가 같으면 0을 반환
    }
    @Override
    public String toString() {
        return this.name + " : " + this.age;
    }
}
```

```
public class ArrayObjSearch {
    public static void main(String[] args) {
        Person[] ar = new Person[3];
        ar[0] = new Person("이", 29);
        ar[1] = new Person("정", 40);
        ar[2] = new Person("허", 15);
        Arrays.sort(ar);
        int idx = Arrays.binarySearch(ar, new Person("Who are
you?", 40));
        System.out.println(ar[idx]);
    }
}
```