로그 파일 재정렬

문제 : 로그를 재정렬하라. 기준은 아래와 같다.
1. 로그의 가장 앞 부분은 식별자로서, 순서에 영향을 끼치지 않는다.
2. 문자로 구성된 로그가 숫자 로그보다 앞에 오며, 문자 로그는 사전순으로 한다.
3. 문자가 동일할 경우에는 식별자순으로 한다.
4. 숫자 로그는 입력 순서대로 한다.

예시1)
Input:["id1 8 1 5 1 ", "id2 art can","id3 3 6", "id4 own kit idg", "id5 art zero"]
Output:["id2 art can", "id5 art zero", "id4 own kit idg", "id1 8 1 5 1 ","id3 3 6"]
예시2)
Input: logs = ["dig1 8 1 5 1","let1 art can","dig2 3 6","let2 own kit dig","let3 art zero"]
Output: ["let1 art can","let3 art zero","let2 own kit dig","dig1 8 1 5 1","dig2 3 6"]

요구 조건을 얼마나 깔끔하게 처리할 수 있는지를 묻는 문제이다.
먼저 문자로 구성된 로그가 숫자 로그보다 이전에 오며, 숫자 로그는 입력 순서대로 둔다.
먼저 로그에서 각각의 종류를 확인하고 문자 로그라면 문자 리스트에, 숫자 로그라면 숫자 리스트에 삽입한다.

```
List<String> letterList = new ArrayList<>();
List<String> digitList = new ArrayList<>();

for (String log : logs){
//isDigit() 함수는 명시된 char 값이 숫자인지 여부를 판단하여 true, false를 반환한다.
        // id 뒤에 오는 문자,숫자를 확인하여 리스트에 삽입한다.
        if (Character.isDigit(log.split(" ")[1].charAt(0))){
                digitList.add(log);
        } else {
                letterList.add(log);
        }
}
```

문자 로그의 정렬 기준은 사전순으로 하되, 문자가 동일할 경우 식별자순으로 한다.

```
Collections.sort(letterList, new Comparator<String>(){
        @Override
        public int compare(String o1, String o2){
                String[] o1x = o1.split(" ", 2);
                String[] o2x = o2.split(" ", 2);
```

```
                int compared = o1x[1].compareTo(o2x[1]);
                if (compared == 0){
                        return o1x[0].compareTo(o2x[0]);
                } else {
                        return compared;
                }
        }
})
```

위 코드에서 사용된 Comparator 알아보자
Comparator는 함수형 인터페이스이다.
Comparator 인터페이스를 구현하려면 compare메서드를 정의해 놓아야하는데, o1, o2 두 값 또는 두 객체를 비교하는 메서드를 재정의 하면된다.
두 요소를 비교해서 o1이 더 작으면 음수, o1과 o2가 같으면 0, o1이 더 크면 양수를 반환하는 메서드를 정의해야한다.

코드에서 `Comparator`는 두 문자열(`o1`, `o2`)을 비교하는 로직을 구현합니다. 각 문자열은 공백으로 구분되어 두 부분으로 나뉩니다(`split(" ", 2)`). 이렇게 분리된 각 부분은 다음과 같이 처리됩니다.

1. **두 번째 부분 비교**: 먼저 두 문자열의 두 번째 부분(`o1x[1]`, `o2x[1]`)을 비교합니다. 이 비교는 문자열의 사전 순서에 따라 수행됩니다. 즉, 알파벳 순서로 누가 앞에 오는지를 결정합니다. (첫번째 부분은 "id"가 포함되어있는 값이다.)
2. **첫 번째 부분 비교**: 만약 두 번째 부분이 같은 경우(즉, `compared == 0`일 경우), 첫 번째 부분(`o1x[0]`, `o2x[0]`)을 비교합니다. 이것도 알파벳 순서에 따라 수행됩니다.

```
// 첫번째
String[] o1x = o1.split(" ", 2);
// o1 : "id4 own kit idg"
// o1x : ["id4", "own kit idg"]
String[] o2x = o2.split(" ", 2);
// o2 : "id2 art can"
// o2x : ["id2", "art can"]

int compared = o1x[1].compareTo(o2x[1]);  // compared = 14;
if (compared == 0){
    return o1x[0].compareTo(o2x[0]);
} else {
    return compared;
}
```

`compareTo` 메서드는 사전적 순서를 기준으로 두 문자열을 비교합니다.
즉, 이 메서드는 두 문자열의 각 문자를 차례대로 비교하면서 두 문자열이 서로 다른 첫 번째 문자에서 ASCII 코

드 값의 차이를 반환합니다.
만약 문자열이 다른 문자열의 접두사인 경우에는 문자열의 길이 차이를 반환합니다
예시)

- 첫 번째 문자열 `"hello"`의 길이는 5입니다.
- 두 번째 문자열 `"hello world"`의 길이는 11입니다.
- 따라서, `compareTo` 메서드는 `5 - 11 = -6`을 반환합니다.

첫 번째 문자열에서 `"o"` (ASCII 코드 111)와 두 번째 문자열에서 `"a"` (ASCII 코드 97)를 비교하면, `111 - 97 = 14`가 됩니다.

이렇게 두 값이 동일하다면 0, 비교 대상의 순서가 앞으로 와할 경우에는 1(양수), 비교 대상의 순서가 뒤에 머물러야 할 경우에는 -1(음수)가 된다.

이제 문자 로그 외에 나머지 숫자 로그를 합치면 된다.
숫자 로그의 경우에는 입력 순서를 그대로 유지한다고 했으니 그냥 두고, 문자 로그의 정렬 결과 뒤에 추가해주기만 하면 된다.

```
public String[] reorderLogFiles(String[] logs){
        // 문자 로그를 저장할 리스트
        List<String> letterList = new ArrayList<>();
        // 숫자 로그를 저장할 리스트
        List<String> digitList = new ArrayList<>();

        for (String log : logs){
                // 로그 종류를 확인 후 숫자 로그라면 숫자 리스트에 삽입
                if (Character.isDigit(log.split(" ")[1].charAt(0))){
                        digitList.add(log);
                } else{
                // 숫자 로그가 아니라면 문자 로그리스트에 삽입
                        letterList.add(log);
                }
        }

        letterList.sort((o1, o2) -> {
                String[] o1x = o1.split(" ", 2);
                String[] o2x = o2.split(" ", 2);

                // 문자 로그 사전순으로 비교
                int compared = o1x[1].compareTo(o2x[1]);
                // 문자가 동일한 경우 식별자 (id)값 비교
                if (compared == 0){
                        return o1x[0].compareTo(o2x[0]);
                } else {
```

```
                        // 비교 대상의 순서가 동일한 경우 0, 순서가 앞인 경우
1(양수), 순서가 뒤인 경우 -1(음수)이 된다.
                        return compared;
                }
        });

        // 문자 리스트 뒤로 숫자 리스트를 이어 붙인다.
        letterList.addAll(digitList);

        // 리스트를 String 배열로 반환한다.
        return letterList.toArray(new String[0]);
}
```