Java Priority Queue

In computer science, a priority queue is an abstract data type which is like a regular queue, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority. - Wikipedia

In this problem we will test your knowledge on Java Priority Queue.

There are a number of students in a school who wait to be served. Two types of events, ENTER and SERVED, can take place which are described below.

- **ENTER**: A student with some priority enters the queue to be served.
- **SERVED**: The student with the highest priority is served (removed) from the queue.

A unique id is assigned to each student entering the queue. The queue serves the students based on the following criteria (priority criteria):

1. The student having the highest Cumulative Grade Point Average (CGPA) is served first.
2. Any students having the same CGPA will be served by name in ascending case-sensitive alphabetical order.
3. Any students having the same CGPA and name will be served in ascending order of the id.

Create the following two classes:

- The **Student** class should implement:
  - The constructor `Student(int id, String name, double cgpa)`.
  - The method `int getId()` to return the id of the student.
  - The method `String getName()` to return the name of the student.
  - The method `double getCGPA()` to return the CGPA of the student.

- The **Priorities** class should implement the method `List<Student> getStudents(List<String> events)` to process all the given events and return all the students yet to be served.

**Input Format**

The first line contains an integer, \( n \), describing the total number of events. Each of the \( n \) subsequent lines will be of the following two forms:

- **ENTER name CGPA id**: The student to be inserted into the priority queue.
- **SERVED**: The highest priority student in the queue was served.

The locked stub code in the editor reads the input and tests the correctness of the Student and Priorities classes implementation.

**Constraints**

- \( 2 \leq n \leq 1000 \)
- \( 0 \leq CGPA \leq 4.00 \)
- \( 1 \leq id \leq 10^5 \)
- \( 2 \leq |name| \leq 30 \)

**Output Format**
The locked stub code prints the names of the students yet to be served. If there are no such student, then the code prints **EMPTY**.

**Sample Input 0**

```
12
ENTER John 3.75 50
ENTER Mark 3.8 24
ENTER Shafaet 3.7 35
SERVED
ENTER Ashley 3.9 42
ENTER Maria 3.6 46
ENTER Anik 3.95 49
ENTER Dan 3.95 50
SERVED
```

**Sample Output 0**

```
Dan
Ashley
Shafaet
Maria
```

**Explanation 0**

In this case, the number of events is 12. Let the name of the queue be $Q$.

- John is added to $Q$. So, it contains $(John, 3.75, 50)$.
- Mark is added to $Q$. So, it contains $(John, 3.75, 50)$ and $(Mark, 3.8, 24)$.
- Shafaet is added to $Q$. So, it contains $(John, 3.75, 50)$, $(Mark, 3.8, 24)$, and $(Shafaet, 3.7, 35)$.
- Mark is served as he has the highest CGPA. So, $Q$ contains $(John, 3.75, 50)$ and $(Shafaet, 3.7, 35)$.
- John is served next as he has the highest CGPA. So, $Q$ contains $(Shafaet, 3.7, 35)$.
- Samiha is added to $Q$. So, it contains $(Shafaet, 3.7, 35)$ and $(Samiha, 3.85, 36)$.
- Samiha is served as she has the highest CGPA. So, $Q$ contains $(Shafaet, 3.7, 35)$.
- Now, four more students are added to $Q$. So, it contains $(Shafaet, 3.7, 35)$, $(Ashley, 3.9, 42)$, $(Maria, 3.6, 46)$, $(Anik, 3.95, 49)$, and $(Dan, 3.95, 50)$.
- Anik is served because though both Anil and Dan have the highest CGPA but Anik comes first when sorted in alphabetic order. So, $Q$ contains $(Dan, 3.95, 50)$, $(Ashley, 3.9, 42)$, $(Shafaet, 3.7, 35)$, and $(Maria, 3.6, 46)$.

As all events are completed, the name of each of the remaining students is printed on a new line.