Problem B. Sorting: Comparator

OS Linux

Comparators are used to compare two objects. In this challenge, you'll create a comparator and use it to sort an array. The *Player* class is provided in the editor below. It has two fields:

1. $name$: a string.
2. $score$: an integer.

Given an array of n *Player* objects, write a comparator that sorts them in order of decreasing score. If 2 or more players have the same score, sort those players alphabetically ascending by name. To do this, you must create a *Checker* class that implements the *Comparator* interface, then write an *int compare(Player a, Player b)* method implementing the [Comparator.compare(T o1, T o2)]() method. In short, when sorting in ascending order, a comparator function returns -1 if $a < b$, 0 if $a = b$, and 1 if $a > b$.

Declare a *Checker* class that implements the *comparator* method as described. It should sort first descending by score, then ascending by name. The code stub reads the input, creates a list of Player objects, uses your method to sort the data, and prints it out properly.

Example
$n = 3$ $data = [[Smith, 20], [Jones, 15], [Jones, 20]]$

Sort the list as $data_{sorted} = [[Jones, 20], [Smith, 20], [Jones, 15]]$. Sort first descending by score, then ascending by name.

Input Format

The first line contains an integer, n, the number of players.
Each of the next n lines contains a player's $name$ and $score$, a string and an integer.

Constraints

- $0 \leq score \leq 1000$
- Two or more players can have the same name.
- Player names consist of lowercase English alphabetic letters.

Output Format

You are not responsible for printing any output to stdout. Locked stub code in *Solution* will instantiate a *Checker* object, use it to sort the *Player* array, and print each sorted element.

Input	Output
`5`	`aleksa 150`
`amy 100`	`amy 100`
`david 100`	`david 100`
`heraldo 50`	`aakansha 75`
`aakansha 75`	`heraldo 50`
`aleksa 150`	

Explanation

The players are first sorted descending by score, then ascending by name.