



ESS 201 Programming II Java

Test 2

23 Sept 2020

*Each question is worth **5 marks for a total of 15 marks**. Please upload the solutions to DomJudge. The submissions will be evaluated for correctness based on DomJudge scoring, as well as review of the code. Please also log into CodeTantra in today's lab meeting, so that we can have communication with the TA's if needed.*

*For each question, you are provided a source file with skeleton code. Complete the code in these files (marked with ...). In general, do not modify the rest of the code, and especially **main** methods, except where marked. The source files are as follows:*

1.  [LeagueTable.java](#)
2.  [Test22.java](#)
3.  [SmallPrinter.java](#)

1. [File LeagueTable.java]

A football league ranks teams in the Standings Table as follows:

- a. The team with the higher points
- b. For teams with the same points, the team that has scored more total goals
- c. if both points and goals are the same, then the teams are listed in increasing alphabetical order

We want to compute the order of the teams in a league by implementing a **Comparator** that codifies the above rules. The rest of the code should not do any comparison, and the ordering should be an outcome of sorting the list of teams and their scores.

To simplify the process, we have a class TeamScore that contains the name of the team, the total points and the total goals scored by each team. Implement this class and the TeamComparator and use these in the main.

The input consists of lines with the Name, Points and Total Goals (String, int, int) as input. An "end" in the first line implies end of input.

Sample input:

```
A 4 6
B 4 7
C 2 6
end
```

Expected output:

```
B
A
C
```

2. [File: Test22.java]

Given a list of integers, we want to iterate through them and process only those integers that are in a “white list” - a list of permitted numbers. We want to structure this so that we can change the *white list* easily and not disturb the rest of the code. Also, we want to encapsulate the iteration process within a customized *iterator* that will skip over values that are **not** in the white list.

We have a class `MyList` that itself contains an `ArrayList` of integers. We would like to implement a special iterator - `WhiteListIterator` - that implements the `Iterator` interface and provides this iteration. That is, successive calls to `next()` of the iterator should return the next element of `MyList` that is also in the white list.

Implement the `MyList` class and the `WhiteListIterator` in the sample code. Follow the outline provided. `MyList` has a method `getIterator` that returns the appropriate `WhiteList` iterator. The main uses this to iterate over the input list, and thus visits only the “permitted” values.

The main has two sample White lists hard coded. For this input:

Expected output:

1
3
9
1
9
11

3. [File SmallPrinter.java]

We have a kind of printer - class SmallPrinter - that implements the Printable interface and is expected to only get text files that do not have the word "junk" in them.

We have a File class that contains the name of the file. We extend this to a TextFile class that also "is a" Printable, and also contains the text contents of the file. The main reads the input, creates TextFiles, and maintains a list of text files that do not contain the word "junk" in them. It then prints each of them by invoking the print method of the Printable interface.

Implement the classes whose outline is provided.

The input is a sequence of pairs of lines, the first contains the file name, and the second line is the "contents" of the text file. Assume that the contents are always only one line. A file name "end" implies end of input.

Sample input:

file 1
good content
file 2
mostly junk
end

Expected Output

file 1
good content