Assignment - Team and Leaves Management

[Weight: 12 marks out of the final mark of this course]

Deadline: April 25, 2016 (Monday of revision week)

For late submissions, 2% of your original marks will be deducted if you hand in 1-day late (i.e. on Apr 26), 4% for 2-days (i.e. on Apr 27), 8% for 3-days (i.e. on Apr 28), 16% for 4-days (i.e. on Apr 29), 32%for 5-days (i.e. on Apr 30), 64% for 6-days (i.e. on May 1). Assignments handed on or after May 1 will get zero mark.

Academic dishonesty is strictly prohibited. The principle concerns whether students get their deserved marks and do not intend to cause unfairness. Dishonesty also involves when one let others have a chance to copy his/her code,

Students must obtain the following results in sequence:

Phase 1 (100% correct in PASS) ==> Phase 2 (100% correct in PASS) ==> Phase 3(100% correct in PASS)

- If you can finish Phase 1 with good programming styles + OO programming skills => up to B-
- If you can finish up to Phase 2 with good programming styles + OO programming skills => up to B+
- If you can finish up to Phase 3 with good programming styles + OO programming skills => up to A-
- If you can finish up to Phase 4 with good programming styles + OO programming skills => up to A+

Each phase has various test cases (eg. Phase 1: 1_a.txt, 1_b.txt, etc..). If you get partial correct, your work is still considered. Eg. If you can pass 1a.txt only, you may get up to C-.

For "Good Programming Styles", note that proper indentations , code-layout formatting, proper , meaningful naming, well-designed classes, methods, fields are more important than writing comments.

Assignment Description

This is a simplified management system for a company to handle the grouping of employees and their annual leaves.

An employee is given an amount of annual leaves (0 - 300 days) upon hired. For simplicity of calculation, we ignore the public holidays and any non-working weekends. That is, for example, a leave period during 20-Mar to 29-Mar is counted as 10 days. A normal employee can take leave very flexibly: no special approval is needed from his boss. As a scenario, Bob and Carol have taken leaves, which are shown in the following listing output:

```
> listLeaves
Bob:
2-Jan-2016 to 8-Jan-2016
Carol:
2-Jan-2016 to 8-Jan-2016
2-Mar-2016 to 9-Mar-2016
```

Each employee can join no team, one team, or more than one teams. Each team has a team head since it is formed. An employee can take multiple roles. As an extreme case, one employee could be the heads of two teams and normal members of other three teams. When we list the members of a team, the head is listed together with normal members, ordered by their names. For example, the following shows the listing of two teams' members:

```
> listTeamMembers
Production Team:
Bob (Head of Team)
Carol

Sales Team:
Ada
Bob (Head of Team)
Tim
```

Actually, being the head of a team has no significant privilege. Even worse, whenever a team's head wants to take leave, he must find somebody from that team to substitute him as the *Acting Head* during the period. Note that if one who takes leave is the head of 2 or more teams, then each team needs an acting head. For example, below shows that Bob takes leaves in 2 periods, and assigns the acting heads for both the Production Team and the Sales Team in each period.

```
> takeLeave|Bob|14-Jan-2016|18-Jan-2016|Production Team|Carol|Sales Team|Tim
Done.

> takeLeave|Bob|02-Jan-2016|04-Jan-2016|Production Team|Carol|Sales Team|Ada
Done.

> listTeamMembers
Production Team:
Bob (Head of Team)
Carol
Acting heads:
2-Jan-2016 to 4-Jan-2016: Carol
14-Jan-2016 to 18-Jan-2016: Carol

Sales Team:
Ada
Bob (Head of Team)
Tim
Acting heads:
2-Jan-2016 to 4-Jan-2016: Ada
14-Jan-2016 to 18-Jan-2016: Tim
```

Your task:

Implement this system based on your learning from CS2312, particularly Lab08, Lab09, and Lab10.

Below are the main requirements and test cases of each phase, and general guidelines. For further details of command formats and required outputs, please refer to the styles in Lab08 Q2 to Lab10 Part 1, and the contents in the given test cases and outputs at the course web.

Phase 1 Hiring employees, creating team, advancement of system date, listing of teams and employees.	Basic testing:	1_a.txt
	Undo/redo:	1_b.txt
	Exceptional cases:	1_c.txt
Phase 2 Leaves taking by normal employees (For this phase, simply assume that they are not heads and not acting heads) , listing of leaves.	Basic testing:	2_a.txt
	Undo/redo:	2_b.txt
	Exceptional cases:	2_c.txt
Phase 3 Adding members to teams, listing of roles and team members.	Basic testing:	3_a1.txt, 3_a2.txt
	Undo/redo:	3_b1.txt, 3_b2.txt
	Exceptional cases:	3_c1.txt, 3_c2.txt
Phase 4 Leave taking by heads: assignment of acting heads. Note 1: Only the team member listing method needs to be revised to show the acting heads. Other listing methods do not need to be revised. Note 2: Acting heads cannot take new leaves which overlap with the acting periods.	Basic testing:	4_a1.txt, 4_a2.txt
	Undo/redo:	4_b1.txt, 4_b2.txt
	Exceptional cases:	4_c1.txt, 4_c2.txt

General guidelines:

Sorting - For listing of teams, sort by the team names
For listing of employees, sort by the employee names
For listing of leaves of an employee, sort by the start day of the leave
For listing of team members, sort by the team names and then the employee names
For listing of roles of an employee, sort by the team names

Ordering and comparison of Days -

To make Day objects comparable, we can simply compare the days as integers like yyyymmdd (eg. 20160305 > 20160301 means 20160305 is later)

Modeling of leave records -

Define a class: `LeaveRecord`, to hold the object fields of the start and end days of each leave. additional field(s) could be added if needed.
An employee may have an `Arraylist` of such leave records.

Listing commands -

Some listing commands may take different counts of arguments. For example, in 2_a.txt, "`listLeaves|Carol`" means to list the leaves of Carol, "`listLeaves`" means to list the leaves of all employees.

Name of source files -

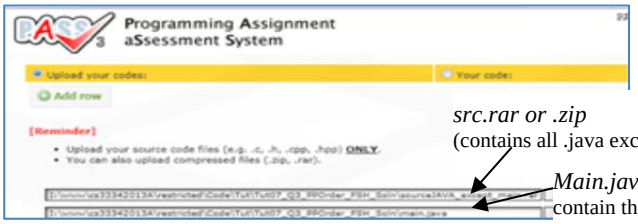
You should name all command classes with the prefix: "`Cmd`", eg. "`class CmdListLeaves`", "`class CmdTakeLeave`"

Handling of errors -

You will need to add handling for many error cases. Most of them should be done by Exception Handling. You should name all Exception classes with prefix: "Ex", eg. "`ExDateHasAlreadyPassed`", "`ExOverlappedLeaves`"

Submission:

Please submit them to PASS as shown below:



src.rar or .zip
(contains all .java except *Main.java*)

Main.java (This class should contain the *main* method)

-- end --