Assignment - Team and Employee Leaves

[Weight: 10 marks out of the final mark of this course]

Deadline: Dec 05, 2025 (Friday)

For late submissions, 20% of your original marks will be deducted if you hand in 1-day late (i.e. on Dec 06), 40% for 2-days (i.e. on Dec 07). Assignments handed on or after Dec 08 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,

Grading: 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) ==> Phase 4

- If you can finish Phase 1 with good programming styles + OO programming skills => up to C+
- 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+

Various test cases are used for each phase (e.g. Phase 1: 1a.txt, 1b.txt, etc..). If you get partial correct, your work is still considered. E.g. 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.
- During marking (Dec 06-24), selected students will be asked to meet me for discussion of your work.

Note:

Please apply what you learn from Lab08, Lab09 - Team Management and Lab10 Q1. You may reuse the code that you work for Lab08 – Lab10. Reusing these code would not be considered as plagiarism.

Please first finish your program for Lab09, then modify and add the required functionalities for this Assignment.

Assignment Description

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

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 extreme cases, one employee could be the heads of two teams and normal members of other three teams, or one employee could be the heads of even more teams and normal members of no (or some) 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 members in the teams, and listing of Bob's roles:

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

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

```
> listAllRoles|Bob
Production Team (Head of Team)
Sales Team (Head of Team)
```

An employee is entitled an amount of *Annual Leaves* (at most 300 days) upon being hired. In addition, there are *Sick Leaves* (maximum 135 days per year) and *No Pay Leave* (no limit). Among the annual leaves, employees should take at least 7 days which are consecutive*.

Therefore, leaves taken would be one of the following types:

(1) AL: Short annual leave (annual leave of 6 days or less)
(2) BL: Block annual leave (annual leave of 7 days or more) *
(3) SL: Sick leave
(4) NL: No pay leave

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.

** This arrangement, also called "Block Leave", is just to take several consecutive days of the entitled annual leave as a block, rather than taking leave one or two days at a time. This is also one of the statutory requirements stated by law in HK: "..At least 7 days should be granted consecutively.. "(https://www.labour.gov.hk/eng/faq/cap57i_whole.htm)*

Also for simplicity, a normal employee can take leave very flexibly: no special approval is needed from his boss. As a scenario, Bob and Carol have taken block annual leave, short annual leave, and sick leave, which are shown in the listing output on the right:

```
> listLeaves
Bob:
2-Jan-2025 to 8-Jan-2025 [BL]
Carol:
3-Jan-2025 to 8-Jan-2025 [AL]
2-Mar-2025 to 2-Mar-2025 [SL]
```

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. E.g., below shows that Bob takes leaves in 2 periods (one annual leave (AL) and one sick leave (SL)), and assigns the acting heads for both the Production and the Sales Team in each period.

```
> applyLeave|Bob|AL|14-Jan-2025|18-Jan-2025|Production Team|Carol|Sales Team|Tim
Done.

> applyLeave|Bob|SL|02-Jan-2025|04-Jan-2025|Production Team|Carol|Sales Team|Ada
Done.

> listTeamMembers
Production Team:
Bob (Head of Team)
Carol
Acting heads:
2-Jan-2025 to 4-Jan-2025: Carol
14-Jan-2025 to 18-Jan-2025: Carol

Sales Team:
Ada
Bob (Head of Team)
Tim
Acting heads:
2-Jan-2025 to 4-Jan-2025: Ada
14-Jan-2025 to 18-Jan-2025: 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, and the contents in the given test cases and outputs for this assignment at the course web.

Phase 1 Hiring employees, creating team, advancement of system date, listing of teams and employees.	Basic testing:	1a.txt
	Undo/redo:	1b.txt
	Exceptional cases:	1c.txt
Phase 2 Adding members to teams, listing of roles and team members. Hint: State Pattern may NOT be needed.	Basic testing:	2a1.txt, 2a2.txt
	Undo/redo:	2b1.txt, 2b2.txt
	Exceptional cases:	2c1.txt, 2c2.txt
Phase 3	Basic testing:	3a1.txt, 3a2.txt

Leaves taking by normal employees (For this phase, simply assume that they are not heads and not acting heads) , listing of leaves.	Undo/redo:	3b1.txt, 3b2.txt
	Exceptional cases:	3c1.txt, 3c2.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:	4a1.txt, 4a2.txt
	Undo/redo:	4b1.txt, 4b2.txt
	Exceptional cases:	4c1.txt, 4c2.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. 20250305 > 20250301 means 20250305 is later)

Modeling of leave records -

Define an abstract 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.

Define new classes under `LeaveRecord` to represent the different types of leaves (AL, BL, SL, NL). You should name these classes using the prefix "`LeaveRecord_`", e.g. "`LeaveRecord_AL`".

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` **`Cmd`**`ListLeaves`", "`class` **`Cmd`**`ApplyLeave`"

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. "**`Ex`**`DateHasAlreadyPassed`", "**`Ex`**`OverlappedLeaves`"

- Please pay attention to the following:

Advantages of Using Exceptions (Week 09 Lecture exercise Q5)
==========
Advantage 1. Separating Error-handling from "Regular" Code (** You should achieve in this assignment (most cases) !!)
Advantage 2. Grouping and Differentiating Errors *(Not needed for assignment)*
Advantage 3. Propagating Erros up the call stack (** MUST achieve in your assignment !!!)

Checking the total count of days of leave –

You will need to check the total count of days of leaves. For example, the total number of days of annual leaves should not exceed the entitled amount. And, if an employee has not taken any *Block Annual Leave* yet, the remaining annual leaves allowed should be at least 7 days so that arranging a block annual leave is still possible.

In reality, leaves should be counted by year, and the entitled maximum leaves should be reset at the beginning of a year period. For simplicity, let's assume that the data handled by the program covers the records from the beginning of one year period, and within one single year period.

Test cases:The requirements in each phase and test cases for reference are given on Page 2-3.

The given test cases and outputs are to show the functionalities that you need to implement. They also serve to specify the input and output formats.

However, note that they do not test rigorously for the accuracy of your program.

For example, your program should be able to count that the leave period from Feb 28, 2024 to Mar 01, 2024 is 3 days (no more, no less). If your program fails to do so, then your grade will be affected due to incorrect solution (despite that you might have obtained 100% correct in PASS).

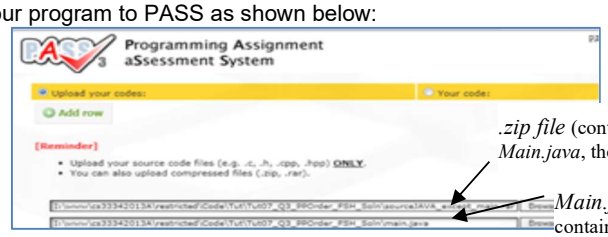
If your program fails to work appropriately, then your grade will be affected due to incorrect solution (despite that you might have obtained 100% correct in PASS).

To help verify that students have applied proper solution design, a few test cases will be used during manual marking. These test cases will be posted after manual marking.

Concerns about efficiency: Assume that we have up to m=5000 employees and n=50000 leave records. For modern computers, keeping these records, doing any linear time operations, and providing sorting/searching are not a problem. Therefore, please spend more time and effort on modelling the entities involved in the case study.

Submission:

Please submit your program to PASS as shown below:



For Mac users,
please read Canvas => CS2312 => Announcements => " Online zip tools for Mac users (for PASS submission)"

-- end --