Assignment - Team and Employee Leaves

[Weight: 15 marks out of the final mark of this course]

Deadline: Dec 08, 2023 (Friday)

For late submissions, 2% of your original marks will be deducted if you hand in 1-day late (i.e. on Dec 09), 25% for 2-days (i.e. on Dec 10), assignments handed in on or after Dec 11 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 (>=90% correct in PASS) ==> Phase 2 (>=85% correct in PASS) ==> Phase 3

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

Various test cases are used for each phase (eg. Phase 1: 1a.txt, 1b.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.
- During marking (Dec 11-24), selected students will be asked to meet me for discussion of your work.

Note:

Please apply what you learn from Lab08 - Lab10. You may reuse the code that you worked 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 system which handles employees in teams and assignment of projects to the teams.

Employees and teams

Each employee is entitled an amount of *Annual Leaves* upon being hired. An employee can join only one team. Each team has a team leader since it is formed. An employee can be the leader of a team or otherwise a normal member. For simplicity we don't handle the cases like an employee leaves a team or changes to another team. Below are the outputs from two listing commands based on the Lab9 program.

```
> listEmployees
Ada (Entitled Annual Leaves: 14 days)
Bob (Entitled Annual Leaves: 21 days)
Carol (Entitled Annual Leaves: 14 days)
Dickson (Entitled Annual Leaves: 21 days)
Emily (Entitled Annual Leaves: 14 days)
```

```
> listTeams
Team Name      Leader    Setup Date
Spider Gang    Bob       1-Feb-2023
Team 007       Carol     1-Feb-2023
```

Taking leaves *(Note: you will handle this in Phase 2)*

Employees can take leaves according to their entitled annual leaves. To calculate leaves in a simple way, we ignore the public holidays and any non-working weekends. That is, for example, a leave period during 2-Mar to 8-Mar is counted as 7 days.

The following is an example of the `takeLeave` command: Bob takes leave for a period of 7 days. The output also indicates that Bob still has 14 days of annual leave remaining.

```
> hire|Bob|21
Done.

> takeLeave|Bob|02-Feb-2023|08-Feb-2023
Done.  Bob's remaining annual leave: 14 days
```

As a further scenario, below shows five employees. Three employees have taken annual leaves. Their leaves are listed using the `listLeaves` command. They have formed 2 teams. The leaves are also listed by teams.

```
> startNewDay|02-Feb-2023
Done.

> listLeaves
Ada: 15-Mar-2023 to 17-Mar-2023
Bob: 2-Feb-2023 to 8-Feb-2023
Carol: 8-Mar-2023 to 12-Mar-2023, 3-Apr-2023 to 8-Apr-2023
Dickson: --
Emily: --

> startNewDay|09-Mar-2023
Done.

> listLeaves
Ada: 15-Mar-2023 to 17-Mar-2023
Bob: --
Carol: 8-Mar-2023 to 12-Mar-2023, 3-Apr-2023 to 8-Apr-2023
Dickson: --
Emily: --

> listLeaves|Ada
Ada: 15-Mar-2023 to 17-Mar-2023
```

The `listLeaves` command lists the current and coming leaves of all employees.

The `listLeaves|[employee name]` command lists those info for one employee.

The `listTeamMembers|[team name]` command lists the head and then members of a team. It also shows their current/coming leaves.

```
> startNewDay|09-Mar-2023
Done.


> setupTeam|Spider Gang|Bob
Done.

> setupTeam|Team 007|Carol
Done.

> joinTeam|Ada|Spider Gang
Done.

> joinTeam|Emily|Spider Gang
Done.

> joinTeam|Dickson|Team 007
Done.

> listTeamMembers|Spider Gang
Role      Name      Current / coming leaves
Leader    Bob       --
Member    Ada       15-Mar-2023 to 17-Mar-2023
Member    Emily     --

> listTeamMembers|Team 007
Role      Name      Current / coming leaves
Leader    Carol     8-Mar-2023 to 12-Mar-2023, 3-Apr-2023 to 8-Apr-2023
Member    Dickson   --
```

Creating projects and assign to teams (*Note: this is included in Phase 1; where each team has a leader only)*

Each project has a code. To create a project, we provide the start day and the length of the project period in days. A project can be assigned to a team only. The example below shows 2 projects, one is assigned to the team `Spider Gang`. The listing of projects shows the day periods as well as the assigned teams and the members of the team (the leader is listed first).

```
> createProject|P345|9-Apr-2023|1
Done.

> createProject|P001|15-Mar-2023|15
Done.

> assign|P001|Spider Gang
Done.

> listProjects
Project  Start Day    End Day      Team
P001     15-Mar-2023  29-Mar-2023  Spider Gang (Bob, Ada, Emily)
P345     9-Apr-2023   9-Apr-2023   --
```

Restriction of leave taking during final stages of projects *(Note: This requirement belongs to Phase 3)*

Each project has a *final stage* during the last few days of the project period. For simplicity we define the last 5 days in a project period as the *final stage* of the project. When an employee wants to take leave, the period must not overlap with any *final stage* of projects that his team works on. For example, the following leave request is not allowed:

```
> listProjects
Project  Start Day    End Day      Team
P001     15-Mar-2023  29-Mar-2023  Spider Gang (Bob, Ada, Emily)
P345     9-Apr-2023   9-Apr-2023   --

> takeLeave|Emily|24-Mar-2023|25-Mar-2023
The leave is invalid.  Reason: Project P001 will be in its final stage during
25-Mar-2023 to 29-Mar-2023.
```

If the period of a project is not more than 5 days, then the whole project period is its *final stage*.

Suggest a team for project assignment *(Note: This requirement belongs to Phase 3)*

For a project *Proj* that is waiting for assigning to any team, we want to find out the most suitable team to take this project. For simplicity, we consider team *T*'s suitability by finding out its *predicted loading factor* (*predicted-lf*) if the team is assigned to take the project.

The calculation for *predicted-lf* is explained below:

1) First, calculate the average manpower (m) of the team *T* during the lifetime of *Proj*.
   For example, suppose there are 2 members in team *T*; but one of them will take leave during half of *Proj*'s lifetime. Then the average manpower will be `m = 1 + 0.5`, so `m = 1.5 workers`.

2) Then, calculate the average project count (p) of team *T* during the lifetime of *Proj*.
   For example, suppose currently *T* is assigned with 3 projects during the period of *Proj*; but one project will finish at the middle of *Proj* 's lifetime. Then the average project count (p) will be `p = 2 + 0.5`, so `p = 2.5 projects`.

3) Finally, calculate the predicted loading factor (*predicted-lf*) for team T based on the above m and p: `predicted-lf = (1 + p) / m`.
   With `m=1.5 workers` and `p=2.5 projects`,

   predicted-lf = (1 + 2.5) / 1.5
   = 2.33 (projects per worker)

A sample output is given below:

```
> suggestProjectTeam|P702
During the period of project P702 (1-Mar-2023 to 30-Mar-2023):
  Average manpower (m) and count of existing projects (p) of each team:
    Team 007: m=3.57 workers, p=1.00 projects
    X Troop: m=2.00 workers, p=0.27 projects
  Projected loading factor when a team takes this project P702:
    Team 007: (p+1)/m = 0.56
    X Troop: (p+1)/m = 0.63
Conclusion: P702 should be assigned to Team 007 for best balancing of loading
```

In case two or more teams are all suitable for the suggestion, please output one of them in the conclusion.

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 a) Hire employee, set up team, start new day; list employees and teams (1a.txt) b) Create project; list projects (1b.txt) c) Assign project to a team; list projects with their teams	Basic testing: Undo/redo: Exceptional cases:	1a.txt, 1b.txt, 1c.txt 1d.txt, 1e.txt, 1f.txt 1g.txt, 1h.txt, 1i.txt
Phase 2 a) Take leave, list current and coming leaves b) Join team, list team members with their leaves	Basic testing: Undo/redo: Exceptional cases:	2a.txt, 2b.txt 2c.txt, 2d.txt 2e.txt, 2f.txt
Phase 3 a) Restriction of leaves during the final stages of projects b) Suggest a team for project assignment	Basic testing and exceptional cases:	3a.txt, 3b.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 days of the leaves
For listing of team members, show the team leader, then the employees by names

Ordering and comparison of Days -
To make Day objects comparable, we can simply compare the days as integers like yyyymmdd (eg. 20220305 > 20220301 means 20220305 is later)

Listing commands -
Some listing commands may take different counts of arguments. For example, in 2a.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`**`TakeLeave`"

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`**`OverlappedLeaves`", "**`Ex`**`EmployeeHasJoinedATeamAlready`"

- 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 !!!)

About checking the total count of days of leave –

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.

About checking of dates –

For simplicity, we assume that when projects are created or leaves are applied, the concerned dates are not earlier than the system date. Also, the formats of the dates are correct and the dates are valid. You do not need to check for these issues.

Test cases:The requirements in each phase and test cases for reference are given on Page 4.

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 on PASS are NOT disclosed. These test cases will be posted after manual marking.

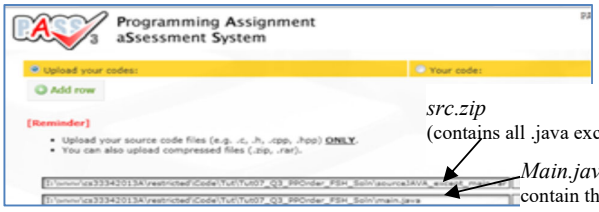
Q: What if my program cannot pass these test cases just because of some minor problems, e.g. spacing or string spelling problems? I don't want to lose my marks just based on these non-technical issues.

A: Helena will manually check the cases and will restore the score for you if the discrepancy is not related to problem solving or code design.

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 them to PASS as shown below:



-- end --