Assignment - Team and Project Management

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

Deadline: December 04, 2020 (Friday of revision week)

For late submissions, 2% of your original marks will be deducted if you hand in 1-day late (i.e. on Dec 05), 25% for 2-days (i.e. on Dec 06), assignments handed in on or after Dec 07 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)

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

Various test cases are used for each phase (e.g. Phase 1: a1.txt, c3.txt, etc..). If you get partial correct, your work is still considered. E.g. If you can pass a1 – b3 only, your grade may be 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 05-19), 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.

I. Assignment Description

This is a simplified management system for a company to handle the working teams, employees and project assignments. The company forms teams to work for projects. Each team has a leader when it is formed. Then other employees may be assigned to join the team as normal team members.

Two listings are shown below. The listing of teams shows the typical information of the teams: a team name, a team leader, the date when the team was setup, and a list of members. The listing of employees shows the teams that the employees belong to, if any.

```
> listTeams
Team Name      Leader    Setup Date    Members
Spider Gang    Brian     15-Feb-2020   Ada Emily
Team 007       Carol     20-Feb-2020   Bill
X Troop        Angel     10-Feb-2020   (no member)
```

```
> listEmployees
Ada (Spider Gang)
Angel (X Troop)
Bill (Team 007)
Brian (Spider Gang)
Carol (Team 007)
Dickson
Emily (Spider Gang)
Helen
```

The listing of projects below shows the typical information of projects: a project has its project code, estimated manpower, the assigned team and the assigned the working period, if any.

```
> listProjects
Project  Est manpower  Team         Start Day    End Day
P001     7 man-days    Spider Gang  28-Feb-2020  2-Mar-2020
P002     10 man-days   (Not Assigned)
P003     10 man-days   (Not Assigned)
```

For simplicity, we have the following assumptions and rules:

- Public holidays and any non-working weekends are ignored. That is, for example, the period during 20-Mar to 29-Mar is counted as 10 working days.

- An employee cannot belong to two teams at the same time. That is, an employee can belong to one team only, either as the leader or a normal member.

- A normal member of a team can change to another team. However, a team leader will never leave his team.

- To run a project, the company first creates the project with the project code and provides an estimation of the manpower needed by the project (in term of a whole number of man-days).

  Then the project is assigned to a team with a start day. The expected end day will be calculated immediately based on the team size (how many members, including the team leader) at the moment when the project is assigned. Fractional parts will be rounded-up. For example, a project of 4 man-days (estimated), if assigned to a team of 3 persons, will take 2 whole days.

  For example, X Troop was assigned to take P002 and then P003:

  P002 (10 man-days) was assigned to X Troop when there was only 1 person (the leader) in X Troop

  P003 (10 man-days) was assigned to X Troop when there were 2 persons in X Troop

```
> listProjects
Project  Est manpower  Team         Start Day    End Day
P001     7 man-days    Spider Gang  28-Feb-2020  1-Mar-2020
P002     10 man-days   X Troop      21-Feb-2020  1-Mar-2020
P003     10 man-days   X Troop      8-Mar-2020   12-Mar-2020
```

- Once assigned, the duration of a project is not supposed to change even if employees are added to or leave the team. *(If a team gets lack of manpower, then everybody needs to work harder (because projects cannot wait), or the management have to hire new employees or to move some existing employees into the team. -- The management will decide. Therefore our program does not need to handle specificially.)*

- Each project is to be assigned to one working team only. Each team can work for at most 1 project at a time. That is, the working periods cannot overlap: the start day of the next project must be later than the end day of its previous project.

  Also, the earliest start day of a project is *tomorrow* (i.e., the next day of the SystemDate). That is, we cannot assign a project immediately on *today* (or any previous day).

II. Basic and Advanced functions

Basic functions (Phase 1-2; Up to ~ A-)

The basic functions that you need to implement are: hire an employee, set up a team, assign an employee to join a team, move an employee to another team, create a project, assign a team to take a project, advance the system date; and list all employees, teams, or projects.
(For details please refer to parts III and IV.)

Advanced functions (Phase 3; Up to ~A+)

Below are two advanced functions that the company needs:

(1) The company often needs to run some urgent projects. Therefore the program needs to suggest the team that can finish the urgent project as early as possible. The output look like:

```
> suggestTeam|P004
Spider Gang (Work period: 3-Mar-2020 to 4-Mar-2020)
```

If two or more teams can give the same earliest completion day, list all of them.

(2) Sometimes the company wants to know what are the teams that an employee has joined (sorted by dates). The company may want to know the worker details of a project. These outputs should look like:

```
> showEmployeeDetails|Bob
The teams that Bob has joined:
Team 007 (21-Feb-2020 to 26-Feb-2020)
Spider Gang (27-Feb-2020 to 10-Mar-2020)
X Troop (12-Mar-2020 to --)
```

```
> showProjectWorkerDetails|P001
Est manpower : 10 man-days
Team         : Spider Gang (Leader is Aaron)
Work period  : 25-Feb-2020 to 27-Feb-2020

Members:
  Ada (25-Feb-2020 to 27-Feb-2020)
  Amy (25-Feb-2020 to 27-Feb-2020)
  Angel (25-Feb-2020 to 27-Feb-2020)
  Bob (27-Feb-2020 to 27-Feb-2020)
```

Note also that if an employee has changed from one team to another team, the program should set his termination date in the previous team to *one day before he changed to the new team*.

(For details please refer to parts III and IV.)

III. General Guidelines:

- The program needs to handle undo-redo of commands.

- Name of source files -
  You should name all command classes with the prefix: "`Cmd`", e.g. "`class CmdJoinTeam`", "`class CmdStartNewDay`"

- Most problem cases should be handled using Exception Handling as covered in this course.
  You should name all Exception classes with prefix: "Ex", e.g. "`ExEmployeeNameAlreadyExists`", "`ExInvalidDate`"

- Please apply the following ordering of listing:

  Listing of employees: order by employee name

  Listing of teams: order by team name

  Listing of projects: order by project code

  Listing of teams that an employee has joined (The advanced function: *showEmployeeDetails*): order by dates.

- Ordering and comparison of Days -
  To make Day objects comparable, we can simply compare the days as integers like yyyymmdd (e.g. 20200305 > 20200301 means 20200305 is later)

- Handling day periods -
  Define a class: `DayPeriod`, to model the day periods, such that each day period object holds the start day and end day. The class should also provide useful methods like checking whether two periods overlap, etc..

- Test cases -
  The given test cases and outputs are to show the functionalities that you need to implement. They also serve as specifications of 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 period of 3 days starting from 28-Feb-2020 is 28-Feb-2020 to 1-Mar-2020 (The end day is not 30-Feb-2020, not 2-Mar-2020 etc.). **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).**

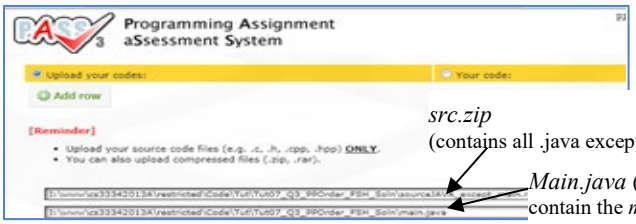
IV. Grading and requirements by phases:

The table below lists the main requirements and test cases of each phase. For command formats and required outputs, please refer to the styles in Lab08 Q2 to Lab10 Q1, and the contents in the given test cases and outputs at the course web.

Phase 1 (a) Hire employee and list employees (b) Setup team, list teams, and start new day (c) Create project and list projects	Basic testing: Undo/redo: Exceptional cases:	a1.txt, b1.txt, c1.txt a2.txt, b2.txt, c2.txt a3.txt, b3.txt, c3.txt
Phase 2 (d) Join team (e) Change team (f) Take project Note: • (d)-(f) You may revise the listing functions for employees, teams, and projects • (f: f4.txt) requires further checking on format of day input using exception handling (e.g. 25-**Fee**-2020, **29**-Feb-2021 are wrong)	Basic testing: Undo/redo: Exceptional cases:	d1.txt, e1.txt, f1.txt d2.txt, e2.txt, f2.txt d3.txt, e3.txt, f3.txt, f4.txt
Phase 3 (g) Suggest the team(s) that can finish a particular project as early as possible (h) Show the teams that an employee has joined (sorted by dates). (i) Show the worker details of a project. Note: For (h) and (i), you need to record the durations which the employees have stayed in various teams. One possible solution is to declare a Membership class and store membership objects as a property in Employee.	Basic testing:	g.txt, h.txt, i.txt

V. Submission:

Please submit your work to PASS as shown below:



-- end --