Design and algorithm

Design:

1. **This project has two classes:**
    a. Class Conference
    b. Class Time
2. **Class Conference:**
    a. **Brief Description:**
        i. The class is used to create a schedule for the selected proposals for a conference.
        ii. It takes selected proposals in an input file, reads the content and stores the data in a datastructure (list of lists).
        iii. **Algorithm:** Schedule selected proposals in sessions to utilise the time window effectively, i.e both the session are tightly packed.
    b. **Attributes:**
        i. **startTime :** Start time for the track.
        ii. **endTime :** End time for the conference OR last time for networking event
        iii. **lunchStartTime :** Time at which lunch will start.
        iv. **lunchEndTime :** Time at which lunch will end and afternoon session will start.
        v. **networkingStartTime :** Time before which networking event can't start.
        vi. All attributes are private.
    c. **Methods**:
        i. **printSchedule()**:
            1. The only **public method** which prints the final schedule of the conference.
            2. The reason why this method is public as the requirement of task is to print the schedule in tracks.
            3. It uses a lot of private methods of the class to achieve the end result.

        4. It hides the background details of event scheduling process.

ii. **__getProposal()**:
   1. **Private method**.
   2. **Tasks performed**:
      a. It reads the selected proposals input file line by line.
      b. Creates a list of lists with Title of the task and time duration in each list.
   3. **Input parameter:** Full path of the input file.
   4. **Return value**: [[‘Talk1 lightning’, 5] , [‘Talk2 30min’, 30]]

iii. **__getEventDuration()**:
   1. **Private method**.
   2. **Tasks performed**:
      a. It takes a line from the file to extract time duration.
   3. **Input parameter**:
      a. A line from the input file.
      b. Example: ‘Talk1 30min’
   4. **Return value**:
      a. Time duration as an integer.

iv. **__getArrangedProposals()**:
   1. **Private method**.
   2. **Tasks performed**:
      a. It takes the list returned from __getProposal() and arrange it as per the algorithm (explained later).
   3. **Return value**:
      a. List of lists arranged as per the algorithm.

v. **__searchComplementaryElements**():
   1. **Private method**.
   2. **Tasks performed**:

                a. It finds other talks which can be grouped together to fit in a span of 60 minutes or less.
                b. It ensures that the time slots are effectively used.
            3. **Input parameter**:
                a. Proposals sorted as per time duration.
                b. Remaining time to complete 60min.
            4. **Return Value**:
                a. Indices of events that can be grouped together.
        vi. **__getSchedule()**:
            1. **Private method**.
            2. **Tasks performed**:
                a. Finalises morning and afternoon sessions with lunch and networking events.
            3. **Output value**:
                a. List of lists with final schedule which can be printed sequentially to print final schedule.
        vii. **__getMessage()**:
            1. **Private Method**.
            2. **Tasks Performed**:
                a. The format in each event is to be printed.
            3. **Input parameter**:
                a. Time of the event (hour and minute)
                b. Name of talk.
            4. **Return value**:
                a. Message (“09:00AM Talk1 30min”)

3. **Class Time**
    a. **Brief Description:**
        i. This class is used primarily to help conference class with various time related operation.
        ii. All the methods in this class are *class methods. .*
    b. **Methods**:
        i. **getMeridem()**:

        1. **Tasks performed**:
            a. Returns correct meridem for given time.
        2. **Input parameter:**
            a. Hour in 24hour clock format.
        3. **Return Value:**
            a. Meridem (AM or PM)
    ii. **clock():**
        1. **Tasks Performed:**
            a. Increments minutes and hours by given minutes.
        2. **Input parameters:**
            a. Hours and minutes.
        3. **Return Value:**
            a. Incremented time.
    iii. **timeFormat():**
        1. **Tasks performed:**
            a. Prints a time in given format as per 12 hour clock.
        2. **Input parameter:**
            a. Hour and minutes.
        3. **Return value:**
            a. Time in expected format.
            b. Example: “09:00AM”
    iv. **convertTo24Hour():**
        1. **Tasks performed:**
            a. Converts 12 hour clock time to 24 hour clock time.
        2. **Input parameters:**
            a. Hour, minutes and meridem.
        3. **Return Value:**
            a. 24 hour clock time.

Algorithm:

1. Initialize start, end, lunch and networking events time.
2. Read input file line by line and extract duration and talk title.

3. Create a list of lists (event, duration).
4. Sort the list in ascending order as per duration.
5. Group events with a total time of 60 minutes or less.
6. Rearrange the list of lists as per new groups identified.
7. Schedule morning and afternoon sessions of the conference in various tracks, if needed.
8. Ensure that each track has a lunch time and a networking event.
9. Print schedule in required format.