Solution 1

```java
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

import java.util.*;

class Program {
  // O(c1 + c2) time | O(c1 + c2) space - where c1 and c2 are the respective numbers of meetings
  // in calendar1 and calendar2
  public static List<StringMeeting> calendarMatching(
      List<StringMeeting> calendar1,
      StringMeeting dailyBounds1,
      List<StringMeeting> calendar2,
      StringMeeting dailyBounds2,
      int meetingDuration) {
    List<Meeting> updatedCalendar1 = updateCalendar(calendar1, dailyBounds1);
    List<Meeting> updatedCalendar2 = updateCalendar(calendar2, dailyBounds2);
    List<Meeting> mergedCalendar = mergeCalendars(updatedCalendar1, updatedCalendar2);
    List<Meeting> flattenedCalendar = flattenCalendar(mergedCalendar);
    return getMatchingAvailabilities(flattenedCalendar, meetingDuration);
  }

  public static List<Meeting> updateCalendar(
      List<StringMeeting> calendar, StringMeeting dailyBounds) {
    List<StringMeeting> updatedCalendar = new ArrayList<StringMeeting>();
    updatedCalendar.add(new StringMeeting("0:00", dailyBounds.start));
    updatedCalendar.addAll(calendar);
    updatedCalendar.add(new StringMeeting(dailyBounds.end, "23:59"));
    List<Meeting> calendarInMinutes = new ArrayList<Meeting>();
    for (int i = 0; i < updatedCalendar.size(); i++) {
      calendarInMinutes.add(
          new Meeting(
              timeToMinutes(updatedCalendar.get(i).start),
              timeToMinutes(updatedCalendar.get(i).end)));
    }
    return calendarInMinutes;
  }

  public static List<Meeting> mergeCalendars(List<Meeting> calendar1, List<Meeting> calendar2) {
    List<Meeting> merged = new ArrayList<Meeting>();
    int i = 0;
    int j = 0;
    while (i < calendar1.size() && j < calendar2.size()) {
      Meeting meeting1 = calendar1.get(i);
      Meeting meeting2 = calendar2.get(j);
      if (meeting1.start < meeting2.start) {
        merged.add(meeting1);
        i++;
      } else {
        merged.add(meeting2);
        j++;
      }
    }
    while (i < calendar1.size()) merged.add(calendar1.get(i++));
    while (j < calendar2.size()) merged.add(calendar2.get(j++));
    return merged;
  }

  public static List<Meeting> flattenCalendar(List<Meeting> calendar) {
    List<Meeting> flattened = new ArrayList<Meeting>();
    flattened.add(calendar.get(0));
    for (int i = 1; i < calendar.size(); i++) {
      Meeting currentMeeting = calendar.get(i);
      Meeting previousMeeting = flattened.get(flattened.size() - 1);
      if (previousMeeting.end >= currentMeeting.start) {
        Meeting newPreviousMeeting =
            new Meeting(previousMeeting.start, Math.max(previousMeeting.end, currentMeeting.end));
        flattened.set(flattened.size() - 1, newPreviousMeeting);
      } else {
        flattened.add(currentMeeting);
      }
    }
    return flattened;
  }

  public static List<StringMeeting> getMatchingAvailabilities(
      List<Meeting> calendar, int meetingDuration) {
    List<Meeting> matchingAvailabilities = new ArrayList<Meeting>();
    for (int i = 1; i < calendar.size(); i++) {
      int start = calendar.get(i - 1).end;
      int end = calendar.get(i).start;
      int availabilityDuration = end - start;
      if (availabilityDuration >= meetingDuration) {
        matchingAvailabilities.add(new Meeting(start, end));
      }
    }
    List<StringMeeting> matchingAvailabilitiesInHours = new ArrayList<StringMeeting>();
    for (int i = 0; i < matchingAvailabilities.size(); i++) {
      matchingAvailabilitiesInHours.add(
          new StringMeeting(
              minutesToTime(matchingAvailabilities.get(i).start),
              minutesToTime(matchingAvailabilities.get(i).end)));
    }
    return matchingAvailabilitiesInHours;
  }

  public static int timeToMinutes(String time) {
    int delimiterPos = time.indexOf(":");
    int hours = Integer.parseInt(time.substring(0, delimiterPos));
    int minutes = Integer.parseInt(time.substring(delimiterPos + 1, time.length()));
    return hours * 60 + minutes;
  }

  public static String minutesToTime(int minutes) {
    int hours = minutes / 60;
    int mins = minutes % 60;
    String hoursString = Integer.toString(hours);
    String minutesString = mins < 10 ? "0" + Integer.toString(mins) : Integer.toString(mins);
    return hoursString + ":" + minutesString;
  }

  static class StringMeeting {
    public String start;
    public String end;

    public StringMeeting(String start, String end) {
      this.start = start;
```

```
      this.end = end;
    }
  }

  static class Meeting {
    public int start;
    public int end;

    public Meeting(int start, int end) {
      this.start = start;
      this.end = end;
    }
  }
}

```