**UMT Software**

**Problema 1**

Szikszai Csaba

**Metoda de rezolvare:**

Am presupus că intervalul calendarul1 și calendarul2 este dat astfel încăt să fie între range1 și range2. Am presupus că Meeinting Time este dat în minute și poate să fie mult de 60 de minute.

La exemplul dat am presupus că primul interval a fost greșit la output și a trebuit să fie ['11:30','12:00'] în loc de ['11:30','12:30']

**Clase:**

**Clasa Calendars:**

Conține un interval de timp prezentat în ore și minte(startHour, startMinute, endHour, endMinute)

Implementează Comparable și suprascrie compareTo pentru a putea sorta după startHour

Suprascrie metoda toString, pentru a putea afișa la final calendarul cerut.

**Clasa People:**

Conține un Obiect Calendars în care păstrăm intervalul range. Conține 2 ArrayLists de Calendars, una pentru calendar cu intervale te timp ocupate și una pentru calendar cu intervale de timp libere.

**Metoda stringToBusyCalendars(String c):**

Transformă un String în ArrayList de Calendars.

**Metoda calendars(String c):**

Transformă un String dat în Obiect de Calendars și la final returnează.

**Metoda** **createFreeCalendar():**

Din obiectul Range și calendarele ocupate ArrayList creează freeCalendar ArrayList. Calendar ocupat fiind sortat, metoda verifică intervalele de timp libere între intervalele date de către busyCalendars și intervalul range.

**Metoda checkTimeInterval(Calendars calendar, int meetingTimeHours, int meetingTimeMinutes):**

Returnează un obiect Calendars, dacă există un interval de timp astfel încât întâlnirea să poată avea loc, în caz contrar returnează null.

**Metoda intersectionOf2Calendars(Calendars calendar1, Calendars calendar2):**

Metoda returnează un obiect Calendars care conține intersecția a două intervale de timp, dacă nu există returnează null. Metoda verifică fiecare caz pentru intersecția intervalelor de timp.

**Metoda intersectionOf2CalendarsList(ArrayList<Calendars> c1, ArrayList<Calendars> c2, int meetingTimeMinutes):**

Metoda returnează un ArrayList de Calendare, care reprezintă intersecții de intervale de timp dintre cele două liste. Metoda verifică dacă meetingTimeMinutes este mai lung de 60 de minute și inițializează meetingTimeHours cu meetingTimeMinutes / 60 și modifică meetingTimeMinutes. Metoda compară fiecare interval de timp dintr-o listă cu fiecare listă de intervale de timp din cealaltă listă și, dacă există o intersecție, astfel încât întâlnirea să poată avea loc, adăugă în listă.

**Clasa Main:**

Citim datele de pe rânduri. Inițializam un obiect People de tipul p1 și p2 cu intervale date folosind metoda calendars(). Inițializam ArrayList busyCalendar pentru p1 și p2 folosind metoda stringToBusyCalendars() și sortăm listele. Aplicam metoda createFreeCalendar() pentru a genera intervale de timp libere pentru p1 si p2. În final, afișăm lista intersecțiilor dintre p1.freeCalendar și p2.freeCalendar prin aplicarea metodei intersectionOf2CalendarsList ().