Solutia Problemei:

- ➔ In prima faza am inceput sa clasific anumite obiecte dupa unele proprietati comune astfel incat sa le pot grupa in clase.
- ➔ Astfel am ajuns la ideea ca un inteval orar ar trebui sa fie un obiect separat(clasa Interval), caracterizat prin doua atribute specifice: ora si minut. Alta clasa relevanta de construit a fost clasa Limit care presupune doua atribute, intervalul orar de start si intervalul orar de final. In final am ajuns la clasa Calendar constituita din doua atribute: o lista de Intervale, alaturi de un alt interval ce reprezinta range-ul din problema.
- ➔ In cadrul implemetarii am incercat sa respect pe cat posibil cele patru principii OOP, principiile SOLID si cele GRASP, astfel incat sa se obtina clase cuplate cat mai slab si care au responsabilitati specifice.
- ➔ Atributele din fiecare clasa sunt declarate private din motivul enuntat anterior, exista gettere si settere pentru accesarea acestora, pentru crearea obiectului exista constructori, pentru afisare s-a suprascris metoda toString astfel sa pot afisa datele dupa cum doresc si sa se apropie cat mai mult de forma data in problema, iar pentru verificarea egalitatii ca continut s-a suprascris metoda equals, unde a fost necesar.
- ➔ Range-ul de lucru pentru ora si minute am presupus ca ar fi [6:00 , 22:00].
- ➔ Am considerat ca datele primite de la utilizator ar fi corecte din acest motiv nu am adaugat verificari in acest sens adica ora va fi o valoare cuprinsa intre 6 si 22 iar minutul va fi o valoare cuprinsa intre 0 si 59.
- ➔ Revenind la solutie, abordarea a pornit de la indeea intervalului fiecarui calendar, daca unul dintre useri este ocupat in perioada/limita specificata, atunci in zona respectiva se va pune in array o valoare diferita de 0 (care inseamna la inceput ideea de liber).
- ➔ Se creeaza un vector de aparitii astfel pentru ca se cunoaste faptul ca valorile vor fi marginite de intervalul 6 – 22. Pentru a putea include si minutele am inmultit ora cu 100 pentru ca sunt doua cifre pentru minut si am adunat ulterior cu valoarea minutului fiecarui interval.
  Astfel se ajunge la un interval constant de aparitii [600 ⇔ 6:00 , 2200 ⇔ 22:00], 2201 deoarece se incepe numerotarea de la 0.
- ➔ Pentru a determina intervalul de lucru am ajuns la ideea ca trebuie luata in calcul doar portiunea determinata de minimul limitei din stanga si minimul limitei din dreapta, pentru ca altfel unul din useri ar fi absent si atunci verificarea nu ar mai avea sens.
- ➔ Pentru a verifica daca timpul in care ambii useri au liber in calendar am facut o transformare din ore in minute pentru a calcula numarul total de minute urmand ca valorile noi calculate sa fie scazute, aceasta metoda apare in clasa Limit, unde mi se pare ca se potriveste cel mai bine.
- ➔ In final pornind de la minimul din stanga catre minimul din dreapta am verificat pozitiile consecutive unde exista valoarea 0 ceea ce inseamna ca acolo ambii useri au liber in calendar, Iar daca numarul acestora este >= decar limita data ca si input, atunci limita de start si cea de final este adaugata la solutie.

  In final se returneaza Calendarul conform caruia ambii useri au liber, iar meet-ul poate avea loc.