Boarding passengers

Proiectul consta in implementarea unui mecanism de ordonare a pasagerilor intr-un avion pentru imbarcare.Acestia detin cate o prioritate in functie de mai multe criterii. Pasagerii se împart în mai multe categorii în funcție de numărul de persoane care călătoresc împreună (familie, grup sau singur) și vârstă (sub 2 ani, între 2-5 ani, între 5-10 ani, între 10-60 ani, peste 60 ani). Biletele de avion sunt de tipurile: Business, Premium și Economic. Compania oferă și alte beneficii precum îmbarcare prioritară și îmbarcare prioritară pentru persoane cu nevoi speciale. Pentru a simplifica procesul de îmbarcare, compania a asociat câte un număr de puncte fiecărui tip de persoană, bilet sau beneficiu. Ordinea îmbarcării se face în funcție de suma obținută de fiecare persoană/familie/grup:

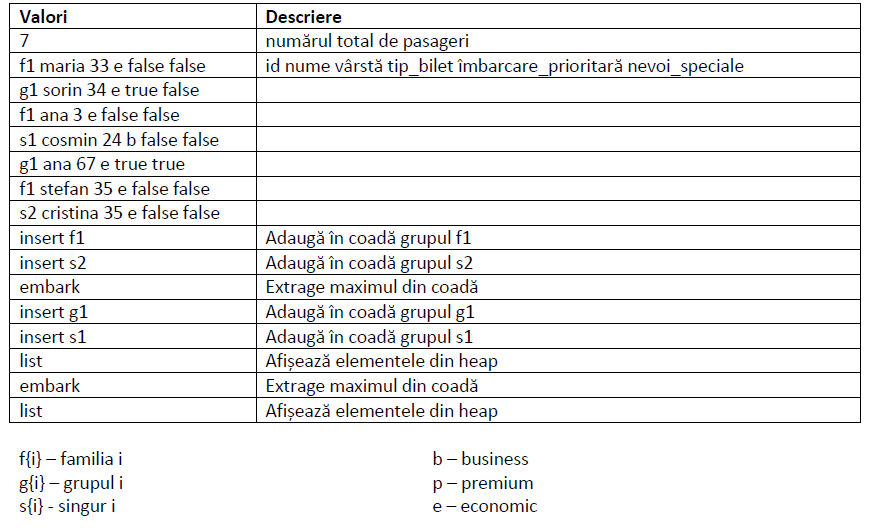
* Familie: 10p
* Grup: 5p
* Singur: 0p
* Sub 2 ani (0 <= age < 2): 20p
* Sub 5 ani (2 <= age < 5): 10p
* Între 5-10 ani (5 <= age < 10 ): 5p
* Între 10-60 ani (10 <= age < 60): 0p
* Peste 60 ani (60 <= age): 15p
* Business: 35p
* Premium: 20p
* Economic: 0p
* Îmbarcare prioritară: 30p
* Nevoi speciale: 100p

Observații:

* În cazul grupurilor și familiilor, persoanele pot avea vârste diferite
* În cazul grupurilor și familiilor, persoanele au același tip de bilet
* În cazul grupurilor și familiilor, persoanele au același tip de îmbarcare (prioritară sau neprioritară);

O coadă de prioritate este o structură de date similară cu o coadă, în care inserarea elementelor se face în funcție de prioritatea elementelor. Astfel, elementele nu se procesează în ordinea în care au fost introduse în coadă, ci în ordinea priorităților, deci este posibil ca un element adăugat la final să fie procesat primul (dacă are prioritatea cea mai mare). O coadă de prioritate se poate implementa în două moduri: printr-o listă sau printr-un heap.

Implementarea folosind heap este cea mai eficientă, având complexitatea O(log N) pentru inserare și ștergere.Mai jos este prezentat fisierul de intare queue.txt:



Se calculeaza prioritatea pentru fiecare entitate:

f1 – 20p

g1 – 280p

s1 – 35p

s2 – 0p

queue.out.txt va contine 2 rânduri:

g1 s2 s1

s1 s2

*Clasa Person.java* modeleaza informatiile despre pasageri.*Clasa Sorting.java* mosteneste Person si implemeteaza interfata Methods ce prezinta metodele de calcul pentru prioritatea fiecarui pasager.Clasa Passenger.java implemeteaza interfata Comparable si modeleaza informatia pastrand prioritatea si cheia pentru fiecare pasager in parte.Clasa PriorityQueue.java realizeaza operatiile propriu-zise pentru mecanism.Putem observa ca functiile de stergere, iserare si listare sunt structurate pe baza unu max-heap(pe langa stegerea (embark), inserarea (insert) si listarea (list) clasa prezinta si metode (sink, swim) ce mentin proprietatea arborelui(complet) construit .Clasa Test.java lucreaza cu fisierele(extragerea inforamtiilor despre pasageri din queue.txt ).De asemenea aici am creat un hasMap realizat pe cheia si prioritatea fiecarui membru.Tot aici am efectuat comenziile din fisier, urmand ca rezultatul lor sa fie scris in fisierul queue.out.txt.