Tema 1

Îmbarcarea pasagerilor în avion

Responsabili: Irina Toma

Introducere

Scopul acestei teme este de a vă familiariza cu programarea orientată pe obiecte și cu limbajul Java. În același timp, vă puteți aprofunda cunoștințele cu privire la o structură de date care este foarte folosită în practică: coada de prioritate (priority queue).

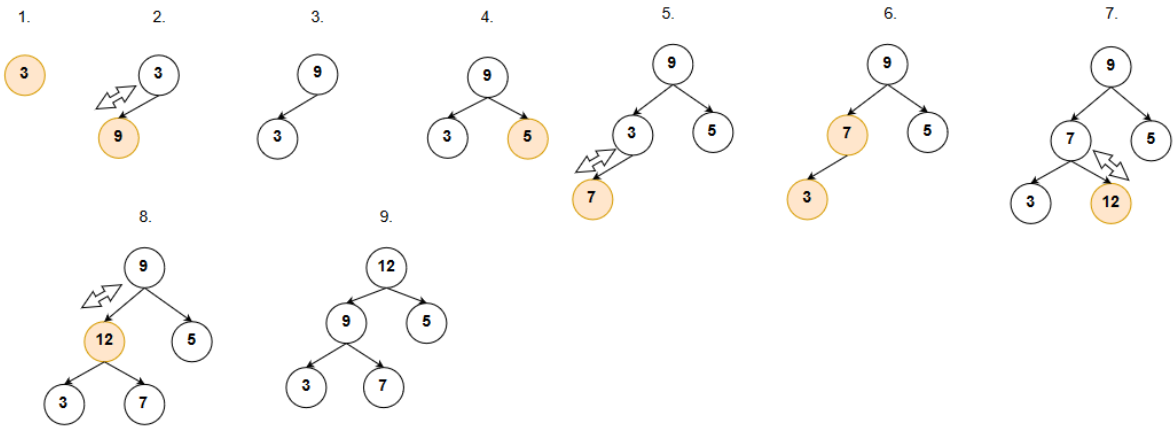
Structura unei cozi de prioritate

O coadă de prioritate [1] 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).

Astfel de structuri se folosesc în implementarea algoritmilor precum Dijkstra, Prim, A* sau pentru compresia datelor (crearea arhivelor de fișiere).

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.

Exemplu:
Considerăm elementele 3, 9, 5, 7, 12. Prioritatea elementelor este dată de valoarea acestora. Adăugarea în heap se face ca în figura de mai jos.



Atenție:

- În heap, fiecare nivel al arborelui, în afară de ultimul, trebuie să fie complet
- Frunzele se adaugă mereu de la stânga la dreapta
- La ștergerea root-ului, ultimul nod adăugat devine root, iar heap-ul se sortează din nou

Descrierea problemei

O companie aeriană vă cere să implementați un mecanism de ordonare a pasagerilor care așteaptă îmbarcarea în avion și vă pune la dispoziție informații despre aceștia: numele, vârsta, tipul de bilet cumpărat și alte beneficii.

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

Exemplu:

- O familie de 3 persoane, cu un copil de 3 ani, fiecare având îmbarcare prioritară obține 10p (familie) + 10p (sub 5 ani) + 3*30p (îmbarcare prioritară) = 110p
- Un grup de 2 persoane la clasa business obține 5p (grup) + 2*35p (business) = 75p
- O familie de 2 persoane, cu îmbarcare prioritară și una dintre persoane având nevoi speciale obține 10p (familie) + 2*30p (îmbarcare prioritară) + 100p (nevoi speciale) = 170p

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)

Testare

Tema va fi testată pe **VMChecker** (va apărea în curând), astfel încât va trebui să aveți și un *makefile* cu o regulă *run* care să ruleze o clasă care conține metoda „main”, pentru a ușura testarea automată.

Fișierul de intrare se va numi *queue.in*, iar cel de ieșire *queue.out*.
Fișierul queue.out trebuie să conțină rezultatul comenzilor *list*, anume parcurgerea heap-ului în preordine la momentul respectiv.

Exemplu:

În queue.in veți primi o serie de comenzi structurate în felul următor: număr pasageri, detalii pasageri, listă comenzi.

Valori	**Descriere**
7	numărul total de pasageri
f1 maria 33 e false false	id nume vârstă tip_bilet îmbarcare_prioritară nevoi_speciale
g1 sorin 34 e true false	
f1 ana 3 e false false	
s1 cosmin 24 b false false	
g1 ana 67 e true true	
f1 stefan 35 e false false	
s2 cristina 35 e false false	
insert f1	Adaugă în coadă grupul f1
insert s2	Adaugă în coadă grupul s2
embark	Extrage maximul din coadă
insert g1	Adaugă în coadă grupul g1
insert s1	Adaugă în coadă grupul s1
list	Afișează elementele din heap
embark	Extrage maximul din coadă
list	Afișează elementele din heap

f{i} – familia i
g{i} – grupul i
s{i} - singur i

b – business
p – premium
e – economic

Se calculează prioritatea pentru fiecare entitate:
f1 – 20p
g1 – 180p
s1 – 35p
s2 – 0p

queue.out va conține 2 rânduri:
g1 s2 s1
s1 s2

Cerințe

1. Să se modeleze conform principiilor programării orientate pe obiecte entitățile din aplicație (de exemplu, Pasager, Bilet, Beneficiu etc.).
    - Trebuie să folosiți conceptele **moștenire, polimorfism** și, opțional, abstractizare
    - Explicați în README alegerile pe care le-ați făcut
    - Pentru o modelare corecta, folosiți ca exemple informațiile prezentate în [2]
2. Să se implementeze o coadă de priorități bazată pe heap (nu aveți voie sa folosiți PriorityQueue din Java). Coada trebuie să implementeze metodele:
    - **insert(Pasager p, int priority)** – adaugă pasagerul P în coada de priorități
    - **embark()** – îmbarcă în avion root-ul heap-ului
    - **list()** – afișează elementele din heap în traversare *preordine (root – left – right)* [3]
3. Să se ruleze linie cu linie comenzile primite în fișierul queue.in
4. Să se afișeze outputul cerut.

Punctaj

- 4.5p teste publice
- 4.5p teste private
- 1p Javadoc
- 2p bonus

Depunctări

- 2p – folosirea structurii PriorityQueue
- 1p – lipsa metodelor insert, embark și list
- 1p – lipsa explicațiilor din README
- Până la 3p pentru modelarea incorectă a entităților din aplicație

Bonus

- 1p – implementarea cozii de prioritate folosind un arbore
- 1p – implementarea metodei delete(Pasager p) care șterge pasagerul asociat din coadă
  Observații:
    - Metoda poate șterge orice pasager, nu doar pe root
    - Metoda poate șterge pasageri singuri, grupuri/familii sau un pasager dintr-un grup/familie
    - La ștergerea unui nod din coada de prioritate, elementul șters este înlocuit cu ultimul element adăugat, apoi se reface ordinea elementelor
    - La ștergerea unui pasager din grup/familie, prioritatea se recalculează și se reface ordinea elementelor din coada de prioritate
    - Fișierul queue.in va conține comenzi de tipul **delete g1** care șterge din coadă grupul g1 sau **delete g1 sorin** care șterge pasagerul cu numele sorin din grupul g1

Referințe

[1] https://www.hackerearth.com/practice/notes/heaps-and-priority-queues/
[2] https://docs.oracle.com/javase/tutorial/java/IandI/index.html
[3] https://www.tutorialspoint.com/data_structures_algorithms/tree_traversal.htm