# Give me more subscriptions

## 1 Introducere

Memoria cache este o colecție de date ce sunt o "copie la indigo" a valorilor originale stocate altundeva sau calculate mai devreme, unde operația de aducere din memorie a datelor originale este costisitoare (datorită timpilor mari de acces la memorie) sau costul recalculării acestora este mare, în comparație cu cel al citirii acestora din cache (o memorie mult mai rapidă și mai ușor de accesat).

Cu alte cuvinte, un cache este o arie temporară de stocare unde datele utilizate în mod frecvent pot fi depozitate pentru un acces rapid la acestea. Odată ce datele sunt stocate în cache, în viitor vor fi luate de aici și utilizate decât să se încerce readucerea datelor originale sau recalcularea acestora, astfel încât timpul mediu de acces este mai mic.

## 2 Cerință

Asa cum a fost mentionat mai sus, dimensiunea unui cache este mult mai mică în comparație cu cea a memoriei principale, iar în cazul în care un nou element se aduce in cache, iar dimensiunea acestuia deja atinge capacitatea maxima, unul din obiectele deja existente in cache va fi inlocuit cu noul element.

### a) Tipuri de cache

Exista diverse metode de a decide ce element va fi eliminat din cache (**numai atunci** cand acesta este plin). In cadrul temei, veti avea de implementat 2 tipuri de cache, ce folosesc 2 metode diferite de stocare a obiectelor:

- **LRU** (least recently used) - elementul cel mai putin utilizat va fi inlocuit cu noul obiect (O modalitate, **care nu e impusa**, de implementare ar fi sa retineti un timestamp dat de un contor care se incrementeaza cand aveti operatii de ADD sau GET, iar la eveacuarea din cache se va alege elementul cu timestamp-ul cel mai mic)
- **FIFO** (first in first out) - elementul de la inceputul cozii va fi eliminat, iar noul element va fi inserat la finalul cozii.

Pentru ambele tipuri de cache se va considera o dimensiune maxima (un numar de obiecte ce pot fi retinute la un moment dat), iar in momentul in care aceasta limita este atinsa, in functie de cache-ul utilizat, la introducerea unui nou obiect se vor folosi regulile mentionate mai sus.

Va trebui sa definiti o interfata "**Cache**" ce declara doua metode: **add** si **remove** (adaugare si eliminare din cache). Ulterior, veti implementa doua clase: **LRUCache** si **FIFOCache**, in care fiecare dintre cele doua metode vor contine implementarea particulara pentru cele doua tipuri de cache.

### b) Memoria principala

Memoria principala va fi reprezentata sub forma unui array ce contine toate obiectele ce vor fi adaugate la un momentdat (se va mentiona in continuarea enuntului cum obiectele sunt introduse in memorie). Chiar daca un obiect se gaseste in cache, acesta va fi prezent si in memoria principala.

### c) Tipuri de obiecte

Obiectele folosite de cele 2 memorii sunt de 3 tipuri. Se va porni de la clasa Subscriptie (clasa abstracta) si vom obtine trei clase: Free (mosteneste clasa abstracta Subscriptie), Basic (mosteneste clasa Free) si Premium (mosteneste clasa Basic). Toate tipurile de subscriptie contin un nume (sir de caractere) ce reprezinta numele titularului ce are subscriptia respectiva.

### d) Subscriptii pentru obiecte

Fiecare subscriptie contine numarul de cereri pana la epuizarea celei de tipul Premium si Basic, iar cea Free este nelimitata. In cazul operatiilor de tip **GET** numele scris este fix tipul subscriptiei (**Premium**, **Basic** sau **Free**). Asta inseamna ca pentru fiecare subscriptie de tip Premium, se va mentiona un numar ce reprezinta numarul maxim de accesari (GET). In momentul in care acest numar este epuizat, subscriptia va deveni automat Basic, analog devenind ulterior Free care poate fi accesata de un numar nelimitat de ori. Pentru constrangeri de implementare legate de modul de utilizare ale acestor subscriptii cititi **sectiunea "4. Constrangeri".**

### e) Operatii

Veti avea de implementat doua operatii de lucru cu memoria. Aceste operatii vor fi mentionate intr-un fisier de intrare (vezi paragraful formatul datelor de intrare/iesire).

- ADD nume_obiect cereri_basic [cereri_premium] - la fiecare adaugare de nou element, acesta va fi inserat numai in memoria principala, nu si in cache, iar in cazul in care un element exista deja in memoria principala sau in ambele, se va suprascrie in memoria principala si va fi eliminat din cache. Parametrul cereri_basic este obligatoriu, iar cereri_premium este optional. Asta inseamna ca pot exista input-uri ca:
  - ADD nume_obiect 3 - ceea ce inseamna ca avem 2 accesari de tip GET basic pentru obiect
  - ADD nume_obiect 3 5 - ceea ce inseamna ca avem mai intai 5 accesari de tip GET premium, urmate de 3 basic.
  - Va trebui sa parsati liniile cu operatii de tipul ADD si sa identificati daca ele contin 1 sau 2 intregi, urmand sa luati decizii ulterior pentru cererile premium/basic.
- GET nume_obiect - aceasta operatie va intoarce un intreg in functie de apartenenta obiectului la cache:
  - 0 - obiectul se gaseste in cache
  - 1 - obiectul se gaseste doar in memoria principala. Ulterior acestei operatii, obiectul va fi mutat si in cache
  - 2 - obiectul nu a fost gasit in memoria principala
  - Alaturi de numarul mentionat mai sus, se va scrie pe aceeasi linie, separat prin spatiu, tipul de subscriptie folosit: "Free", "Basic" sau "Premium". Pentru obiectele care nu se gasesc in memoria principala (intorc 2 la rezultatul GET), singurul lucru printat va fi numarul 2.

### f) Bonus

Se pot acorda pana la 2 puncte bonus pentru implementarea unui al treilea tip de cache, numit Least Frequently Used (LFU). Politica de eliminare din cache pentru LFU este urmatoarea:

- de fiecare data cand un obiect trebuie eliminat din cache, se va elimina cel care a fost **accesat de cele mai putine ori** de cand se afla in cache
- daca toate obiectele care se afla in cache la un momentdat au fost accesate de un numar de ori egal, atunci cel care va fi eliminat va fi primul care a fost adaugat (fiind cel mai vechi).

### g) Formatul datelor de intrare/iesire

Datele de intrare se vor citi din fisierul "configuratie.txt". Este garantat faptul ca datele de intrare vor avea mereu acest format.

- Pe prima linie se va gasi un string reprezentand tipul de cache folosit: "**LRU**" sau "**FIFO**". Pentru bonusul de 2 puncte, pe prima linie se va gasi "**LFU**".
- Pe a doua linie veti gasi numarul maxim de obiecte ce pot fi stocate in cache
- Pe a treia linie veti gasi un numar N (numar strict pozitiv) ce semnfica numarul de operatii (ADD + GET)
- Pe urmatoarele N linii veti gasi operatiile de efectuat.

Datele de iesire se vor scrie in fisierul "out.txt". Acesta va contine M linii, unde M reprezinta numarul de operatii de tip GET din cele N aflate in fisierul de intrare. Fiecare linie va contine un numar intreg (0, 1 sau 2 si un string aferent subscriptiei), asa cum a fost mentionat la punctul anterior.

**Exemplu**

| configuratie.txt | out.txt |
|---|---|
| FIFO<br>2<br>6<br>ADD car 2 1<br>ADD ball 1<br>GET car<br>GET ball<br>GET table<br>GET car | 1 Premium<br>1 Basic<br>2<br>0 Basic |

Dupa cum se observa, pentru primul GET, obiectul “car” se afla in memoria principala (nu a mai fost accesat) si are o cerere de tip premium ce poate fi epuizata (2 basic, 1 premium). Analog obiectul “ball”, insa pentru care se incepe cu o cere Basic. Obiectul “table” nu se gaseste in memoria principala, vom printa 2. La o noua accesare a obiectului “car”, acesta se afla in cache (vom intoarce 0), dar de asemenea a fost decrementat la o cere Basic (cea Premium a fost epuizata la GET-ul anterior).

# 3 Testare

Tema va fi testata si verificata folosind vmchecker.

In unele situatii punctajul de pe vmchecker (cel aferent testelor) poate fi diferit de nota finala acordata de corector (nu ati respectat principiile de OOP sau cele de la sectiunea constrangeri, ati scris codul intr-un singur fisier etc).

# 4 Constrângeri

In cazul in care numarul de subscriptii Premium este egal cu 0 nu este permisa crearea unui obiect Basic, trebuie pastrat acelasi obiect, decrementarea pentru subscriptia Basic realizandu-se prin urcarea in lantul de mosteniri pana cand se ajunge la subscriptia Free care este nelimitata (ajungere la Free daca este cazul - s-a epuizat contorul Basic).

# 5 Punctaj

Punctajul pe tema va consta din:

- 90% testare automată
- 10% coding style + **readme** + JavaDoc
- Code inspection (**-100%**)

### Mentiuni:

- Fisierele de input si output vor fi pasate in **linie de comanda** si se va lucra cu vectorul argv pentru citirea parametrilor.
- Clasa principala a temei se va numi obligatoriu **Main**.
- Tema va fi incarcata atat pe vmchecker, cat si pe site-ul de curs, cu o arhiva ce va avea numele de tipul: Nume_Prenume_Grupa_Tema1POO (Exemplu: Popescu_Ion_322CB_Tema1POO).
- Toate fisierele incluse in arhiva se vor afla la radacina acesteia. Arhiva nu va contine directoare separate in care sa fie puse fisierele .java.