Tema 2 POO (2022 - 2023)

Responsabili: Andreea Duțulescu, Laur Neagu

Introducere

Pentru a ajuta la digitalizarea proceselor din instituțiile publice, ești rugat să implementezi un sistem de management al unei primării. Principala provocare este gestionarea cererilor cetățenilor de către funcționarii publici. Pentru a evita cozile pentru înregistrarea unei afaceri, schimbarea actelor de identitate sau înregistrarea legitimației de student, administrația a decis implementarea unui sistem informatic care să gestioneze aceste activități.

Funcționalități

Cum majoritatea cetățenilor au cereri pentru schimbarea de documente sau anumite task-uri administrative, se dorește gestionarea cât mai eficientă a acestora.
O **Cerere** este definită prin conținutul text al acesteia (de exemplu, *Subsemnatul X, solicită Y*), data la care a fost făcută (zz-LLL-aaa HH:mm:ss) [1] și prioritatea acesteia (un număr de la 1 la 5, 5 fiind prioritatea cea mai urgentă). Se garantează faptul că momentele de timp la care sunt făcute cererile sunt unice. În cadrul modelării cererii trebuie să fie definită și o enumerare internă [2] pentru diferitele tipuri de cereri, cu valori pentru:

```
inlocuire buletin
inregistrare venit salarial
inlocuire carnet de sofer
inlocuire carnet de elev
creare act constitutiv
reinnoire autorizatie
inregistrare cupoane de pensie
```

Enumerarea va fi folosita pentru construirea textului cererii. Conținutul text al cererii depinde de tipul cererii și de persoana care înaintează cererea.

Sistemul admite diferite tipuri de utilizatori. Fiecare utilizator trebuie să îndeplinească anumite caracteristici, modelate printr-o interfață sau clasă abstractă **Utilizator**, detaliată mai jos. Tipurile posibile de utilizatori sunt: **Persoana** (persoană fără o activitate, definită prin `nume`), **Angajat** (persoană angajată, definită prin `nume` și `compania` la care lucrează), **Pensionar** (definit prin `nume`), **Elev** (definit prin `nume` și `școala` la care studiază) și **EntitateJuridica** (o companie, definită prin `nume` și `reprezentant`). Toate câmpurile menționate sunt de tipul **String** și fiecare entitate trebuie să admită un constructor pentru setarea parametrilor.

Funcționalitățile unui Utilizator

Scrierea textului unei cereri (primeste ca parametru tipul cererii (o valoare din enumerare), returneaza un **String** - În funcție de tipul de cerere dorit și al informațiilor de identificare a utilizatorului, metoda trebuie să genereze textul cererii pe baza unor template-uri specifice fiecărui tip de Utilizator. Dacă un tip de cerere nu se pretează unui anumit utilizator (de exemplu, un Pensionar nu poate solicita înlocuirea carnetului de elev), metoda va rezolta într-o excepție modelată de voi care urmează să fie tratată.

Utilizator	Template pentru textul cererii	Tipuri de cereri posibile
Persoana	`Subsemnatul <nume>, va rog sa-mi aprobati urmatoarea solicitare: <tip_cerere>` ex: `Subsemnatul Ion Popescu, va rog sa-mi aprobati urmatoarea solicitare: inlocuire buletin`	`inlocuire buletin` `inlocuire carnet de sofer`

Angajat	`Subsemnatul <nume>, angajat la compania <companie>, va rog sa-mi aprobati urmatoarea solicitare: <tip_cerere>`	`inlocuire buletin`
		`inlocuire carnet de sofer`
		`inregistrare venit salarial`

Pensionar	`Subsemnatul <nume>, va rog sa-mi aprobati urmatoarea solicitare: <tip_cerere>`	`inlocuire buletin`
		`inlocuire carnet de sofer`
		`inregistrare cupoane de pensie`

Elev	`Subsemnatul <nume>, elev la scoala <scoala>, va rog sa-mi aprobati urmatoarea solicitare: <tip_cerere>`	`inlocuire buletin` `inlocuire carnet de elev`
EntitateJuridica	`Subsemnatul <reprezentant>, reprezentant legal al companiei <companie>, va rog sa-mi aprobati urmatoarea solicitare: <tip_cerere>`	`creare act constitutiv` `reinnoire autorizatie`

Crearea unei cereri date fiind tipul acesteia, prioritatea și data la care a fost făcută
Retragerea unei cereri făcute la un moment de timp specificat
Mentinerea unor colecții cu cererile soluționate sau în așteptare
Afișarea cererilor soluționate, in ordinea datei de creare
Afișarea cererilor în așteptare, in ordinea datei de creare

Funcționalitățile unui Birou

Primăria are mai multe birouri. Un **Birou** este specializat pe un anumit tip de cetățean și nu poate primi cererile altor tipuri de utilizatori (acest lucru trebuie modelat prin genericitate [3]). În plus, este evident că un birou nu ar trebui să poată primi cererile unei alte entități care nu este **Utilizator**. Biroul are o colecție de cereri făcute de cetățeni spre a fi soluționate. Totuși, aceste cereri nu sunt soluționate pe baza primul-venit-primul-servit, ci în primul rând în funcție de prioritatea lor, iar apoi, la priorități egale, în funcție de data la care au fost făcute. În vârful cozii de cereri trebuie să se afle cererile cu prioritate mare și cele vechi. Pentru fiecare **Cerere**, în cadrul biroului trebuie să se cunoască utilizatorul (ca referință spre obiect) care a efectuat cererea. Biroul poate primi o cerere și o poate adăuga în colecția lui (se primește obiectul de tip **Cerere** și obiectul de tip **Utilizator**, corespunzător cu tipul acceptat).

Funcționalitățile unui FunctionarPublic

În cadrul unui birou lucrează un număr de funcționari publici, menținut ca o colecție. Aceștia sunt cei care pot lua cererile din coadă și le pot soluționa (evident, în ordinea în care dictează prioritatea și data). Entitatea de **FunctionarPublic** este definită prin `nume` (**String**). Evident, toți funcționarii din cadrul unui birou pot soluționa doar cererile făcute de tipul de utilizator pe care biroul s-a specializat. Astfel, funcționarii pot procesa cererile unui singur tip de utilizator, acesta fiind specificat la adăugarea unui nou funcționar, ca în tabelul de mai jos. Pentru a ține evidența offline a activității funcționarilor, imediat ce un funcționar acceptă o cerere din birou și o

soluționează, sistemul notează informațiile despre cerere într-un fișier special cu numele `functionar_<nume_functionar>.txt`. Aici vor fi scrise cererile, fiecare pe câte un rând, în ordinea soluționării lor, cu următorul format:
`<data_inregistrarii> - <text_cerere>`

Funcționalitățile în ManagementPrimarie

În final, aceste entități vor fi coordonate în **ManagementPrimarie**, care conține următoarele informații:

- câte un birou specializat pentru fiecare tip de utilizator
- o colecție cu toți utilizatorii existenți în sistem

Funcția **main** din cadrul acestei clase primește ca argument în linia de comandă un nume de fișier de intrare care conține informațiile pe care le primește sistemul de la frontend. Informațiile vor fi date câte una pe linie, procesate în ordinea în care apar și au următorul format:

`adauga_functionar; <tip_utilizator/birou>; <nume_functionar>`
 	
`ex: adauga_functionar; elev; Ion Popescu`
	Adaugă un funcționar în cadrul biroului potrivit. Acest funcționar va putea procesa doar cererile tipului de utilizator specificat (în cazul exemplului, **Elev**)

`adauga_utilizator; <tip_utilizator>; <nume_utilizator>`
 	
`ex: adauga_utilizator; entitate juridica; Ioana Ionescu`
	Adaugă un utilizator de un anumit tip în cadrul sistemului de management al primăriei.

`cerere_noua; <nume_utilizator>; <tip_cerere>; <data>; <prioritate>`
 	
`ex: cerere_noua; George Georgescu; inlocuire carnet de sofer; 12-Dec-1999 12:00:01; 4`
	O cerere nouă este creată de un utilizator. Se garantează că utilizatorul există în sistem. Primul pas este redactarea textului cererii pe baza template-urilor date. În cazul în care tipul cererii nu se pretează pentru tipul utilizatorului, funcția din **Utilizator** de scriere a unei cereri va arunca o excepție. Excepția va fi tratată în cadrul funcției de creare a unei cereri, când se va afișa mesajul `Utilizatorul de tip <tip_utilizator> nu poate inainta o cerere de tip <tip_cerere>`
	Ex: `Utilizatorul de tip angajat nu poate inainta o cerere de tip inlocuire carnet de elev`
	După afișarea mesajului, cererea nu se va înregistra și programul va continua. În cazul unei reușite, cererea se va adăuga în cadrul cererilor în așteptare ale utilizatorului și în cadrul colecției ordonate de cereri a biroului corespunzător.

`retrage_cerere; <nume_utilizator>; <data>`
 	
`ex: retrage_cerere; Ioana Ionescu; 12-Jan-2022 00:00:01`
	Cererea specifică datei menționate este retrasă, atât din cererile în așteptare ale utilizatorului, cât și din cadrul biroului corespunzător.

`rezolva_cerere; <tip_utilizator/birou>; <nume_functionar>`	Funcționarul al cărui nume este specificat, alege să rezolve cererea cea mai urgentă din cadrul biroului său. Astfel, cererea va fi

`ex: rezolva_cerere; pensionar; Ion Popescu`	scoasă din coada biroului. În cadrul utilizatorului care a făcut cererea, aceasta este tratată ca nemaifiind în așteptare, devenind finalizată. Funcționarul rezolvă cererea instant și o scrie în fișierul corespunzător numelui său.

`afiseaza_cereri_in_asteptare; <nume_utilizator>`
 	
`ex: afiseaza_cereri_in_asteptare; Ioana Ionescu`
	Afișarea cererilor în așteptare, câte una pe linie, ale unui utilizator, în ordine cronologică (cea mai recentă cerere la final).
	`<data_inregistrarii> - <text_cerere>`
	Ex: `12-Dec-2022 12:00:00 - Subsemn…`

`afiseaza_cereri_finalizate; <nume_utilizator>`
 	
`ex: afiseaza_cereri_finalizate; Ioana Ionescu`
	Afișarea cererilor finalizate, în același format ca mai sus.

`afiseaza_cereri; <tip_utilizator/birou>`
 	
`ex: afiseaza_cereri; angajat`
	Afișarea cererilor existente în cadrul unui birou, în ordinea corespunzătoare priorității și a datei.
	`<prioritate> - <data_inregistrarii> - <text_cerere>`
	Ex: `5 - 10-Jan-2022 10:04:00 - Subsemn…`

Input și Output

Funcția main va primi ca argument în linia de comandă denumirea fișierului de test. Fișierele de input și output au ambele această denumire. Fișierele de input se găsesc în directorul `resources/input/`, iar cele de output trebuie create în `resources/output/`. Fișierele de output ale comenzilor vor avea denumirea dată ca argument, iar cele specifice funcționarilor vor avea denumirea menționată în enunț. Nu modificați fișierele de referință!
La începutul rulării, folderul de output va fi golit automat în primul test.

Exemplu: Funcția main primește ca parametru `"test1.txt"`. Fișierul din care se va face citirea comenzilor se afla la `resources/input/test1.txt`. Fișierul în care se va scrie outputul comenzilor de afișare va fi `resources/output/test1.txt`. Fișierul corespunzător unui posibil funcționar în cadrul testului se va afla la `resources/output/functionar_Nume.txt`.

Punctaj

*Se cere folosirea exclusivă a Java Collections. Nu se admit tablouri standard.

60 de puncte - Suita de teste
20 de puncte - Folosirea colecțiilor eficiente din punct de vedere al complexității de execuție, potrivite pentru operațiile efectuate
10 puncte - Folosirea genericității în cadrul claselor Birou și FuncționarPublic
5 puncte - Folosirea conceptelor de POO studiate (encapsulare, abstractizare, moștenire și polimorfism)
5 puncte - Readme detaliat în care să se specifice motivul alegerii unui anumit tip de colecție pentru un anume task

10 puncte - Bonus: Primăria dorește să mențină în sistemul său evenimentele organizate în oraș. Task-ul bonus implică design-ul și implementarea acestui sistem și oferirea unor fișiere de test cu comenzi pentru testarea funcționalității. Este la latitudinea voastră să alegeți ce comportament și scop va avea această componentă.

[1] https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html
[2] https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html
[3] https://docs.oracle.com/javase/tutorial/java/generics/types.html