Proiect POO 2026

Aplicație de management al centrelor de date

Responsabil: Ilie Cristian Dorobăț
Deadline: 02 februarie, ora 22.00
Assignment: https://classroom.github.com/a/r1iBXMxX
Template: https://github.com/ACS-POO-2CB/proiect-2026-template -> dacă intervin corectări ale testelor de aici vă puteți lua varianta actualizată.
Versiune document: update 1

Introducere

Scopul acestui proiect îl reprezintă dezvoltarea unui API care să permită gestionarea alertelor de sistem. Aplicația are la bază o componentă de management al infrastructurii IT, una pentru managementul grupurilor de monitorizare și alta pentru generarea efectivă a alertelor de sistem. Toate acestea sunt interconectate prin intermediul adresei IP a serverelor din baza de date.
Aplicația va procesa date din fișiere de input și va simula comportamentul unui sistem real de monitorizare al infrastructurii.

Entități

Pentru atingerea obiectului acestui proiect, aveți nevoie de crearea unor entități obligatorii, la care se pot adăuga, după caz, entități opționale structurate de voi. Entitățile obligatorii sunt definite mai jos.

Enumerări

Clasă	Atribute

`PathType`	`GROUPS`
	`LISTENER`
	`SERVERS`

`AlertType`	`ANOMALY` `ADVISORY`

`Severity`	`LOW`
	`MEDIUM`
	`HIGH`
	`CRITICAL`

`ServerStatus`	`UP`
	`DOWN`
	`DEGRADED`

Entități pentru gestionara operatorilor

User

Clasa `User` este **clasa de bază** folosită pentru definirea utilizatorilor care interacționează cu sistemul. Această entitate conține următoarele atribute:

- `String name`
- `String role`
- `String email`

Singurul constructor expus toți parametrii.

Operator

Clasa `Operator` este o **clasă specifică** folosită pentru **definirea efectivă a operatorilor umani**. Aceasta moștenește atributele și metodele din clasa `User`, la care mai adaugă următorul atribut suplimentar:

- `String department`

Constructorul acesteia are semnătura similară cu cea a clasei `User`.

Admin

Clasa `Admin` este, de asemenea, o **clasă specifică** prin intermediul căreia sunt definiți utilizatorii cu drepturi de administrare. Aceasta **moștenește** atributele și metodele clasei `Operator`, iar pe lângă acestea mai este adăugat următorul atribut:

- `int clearanceLevel`

Și în acest caz, constructorul este similar cu cel al clasei `User`.

Entități pentru gestionarea infrastructurii

Server

Clasa `Server` descrie un server monitorizat. Această clasă cuprinde următoarele câmpuri obligatorii:String ipAddress

- `Location location`
- `User owner`

și următoarele câmpuri opționale:

- `String hostname`
- `ServerStatus status`
- `Integer cpuCores`
- `Integer ramGb`
- `Integer storageGb`

Location

Clasa `Location` cuprinde următoarele câmpuri obligatorii:

- `String country`

și următoarele câmpuri opționale:

- `String city`
- `String address`

- `Double latitude`
- `Double longitude`

Entități pentru grupuri de monitorizare

ResourceGroup

Clasa `ResourceGroup` este folosită pentru definirea grupurilor responsabile pentru monitorizarea resurselor. Această entitate conține următoarele atribute:

- `List<User> members`
- `String ipAddress`

și metodele următoare:

- `addMember` - adaugă un singur membru al grupului
- `getMembers` - extrage lista de membri ai grupului
- `removeMember` - elimină un membru din grup

Entități pentru gestionarea alertelor

Alert

Clasa `Alert` este folosită pentru definirea notificărilor de sistem. Aceasta cuprinde următoarele atribute:

- `AlertType type (ANOMALY / ADVISORY)`
- `Severity severity`
- `String message`
- `String ipAddress`

Entități de manipulare a bazei de date

Database

Clasa `Database` simulează interacțiunea cu baza de date și trebuie implementată folosind un design pattern care previne instanțierea multiplă. Aceasta cuprinde următoarele atribute:

- `Database instance` – instanța unică a bazei de date
- `Set<Server> servers` – colecția de servere
- `Set<ResourceGroup> resourceGroups` – grupuri de monitorizare
- `Set<Alert> alerts` – alertele generate în sistem

și următoarele metodele:

- `addServer` - adaugă un singur server
- `addServers` - adaugă o colecție întreagă de servere
- `addResourceGroup` - adaugă un singur grup de monitorizare
- `addResourceGroups` - adaugă o colecție de grupuri de monitorizare
- `addAlert` - adaugă o alertă

Entități pentru gestionarea excepțiilor

Denumire	Mesaj
`LocationException`	LocationException: Country is missing.
`MissingIpAddressException`	MissingIpAddressException: Server IP Address was not provided.
`UserException`	UserException: Name and role can't be empty.

Funcționalități și comenzi

***Notă: <params>** reprezintă lista de parametri ai comenzii. Această listă coincide cu header-ul fișierului de input.*

Adăugarea serverelor

Serverele (instanțele clasei `Server`) trebuie create în mod dinamic folosind fișierele de input `servers_xx.in`, unde `xx` reprezintă un index format din două numere (ex.: 01, 02, etc.).

ADD SERVER <params>

Comanda este folosită pentru adăugarea unui server. În cazul procesării cu succes, în fișierul de output va fi scris următorul mesaj:

- `ADD SERVER: <ipAddress>: <serverStatus>`

Operațiunea poate genera următoarele excepții:

- Adresa IP lipsește:
  - `ADD SERVER: MissingIpAddressException: Server IP Address was not provided. ## line no: <lineNumber>`
- Numele sau rolul owner-ului nu au fost furnizați:
  - `ADD SERVER: UserException: Name and role can't be empty. ## line no: <lineNumber>`
- Numele țării în care este localizat serverul nu există:
  - `ADD SERVER: ADD SERVER: LocationException: Country is missing. ## line no: <lineNumber>`

Gestionarea grupurilor de monitorizare

Grupurile de monitorizare (instanțele clasei `ResourceGroup`) trebuie create în mod dinamic folosind fișierele de input `groups_xx.in`, unde `xx` reprezintă un index format din două numere (ex.: 01, 02, etc.).

Notă**: Pentru comenzile destinate adăugării, căutării și eliminării membrilor din grupurile de monitorizare, mai întâi acest grup trebuie identificat folosind adresei IP a serverului asociat. **În continuare ne vom referi la acest grup ca fiind GRUPUL ȚINTĂ.

ADD GROUP <params>

Comanda este folosită pentru crearea unui grup de monitorizare asociat unui server prin intermediul adresei IP. În cazul procesării cu succes, va fi creat un nou grup, iar următorul mesaj va fi scris în fișierul de output:

- `ADD GROUP: <ipAddress>`

În cazul în care adresa IP lipsește, va fi aruncată o eroare de tipul `MissingIpAddressException`, având următorul mesaj:

- `ADD GROUP: MissingIpAddressException: Server IP Address was not provided. ## line no: <lineNumber>`

FIND GROUP <params>

Comanda permite căutarea unui grup pe baza adresei IP a serverului. În cazul procesării cu succes, în fișierul de output vor fi scrise următoarele mesaje, în funcție de context:

- Grupul a fost identificat pe baza adresei IP:
  - `FIND GROUP: <ipAddress>`
- Grupul nu a putut fi identificat:
  - `FIND GROUP: Group not found: ipAddress = <ipAddress>`

În cazul în care adresa IP lipsește, va fi aruncată o eroare de tipul `MissingIpAddressException`, având următorul mesaj:

- `FIND GROUP: MissingIpAddressException: Server IP Address was not provided. ## line no: <lineNumber>`

REMOVE GROUP <params>

Comanda este utilizată pentru ștergerea unui grup pe baza adresei IP a serverului. În cazul procesării cu succes, în fișierul de output vor fi scrise următoarele mesaje, în funcție de context:

- Grupul a fost identificat pe baza adresei IP:
  - `REMOVE GROUP: <ipAddress>`
- Grupul nu a putut fi identificat:
  - `REMOVE GROUP: Group not found: ipAddress = <ipAddress>`

În cazul în care adresa IP lipsește, va fi aruncată o eroare de tipul `MissingIpAddressException`, având următorul mesaj:

- `REMOVE GROUP: MissingIpAddressException: Server IP Address was not provided. ## line no: <lineNumber>`

ADD MEMBER <params>

Comanda permite adăugarea unui membru (membrii grupului pot fi atât operatori cât și administratori). În cazul procesării cu succes, va fi adăugat un nou membru în **GRUPUL ȚINTĂ**, iar următorul mesaj va fi scris în fișierul de output:

- `ADD MEMBER: <ipAddress>: name = <name> && role = <role>`

În cazul în care adresa IP lipsește, va fi aruncată o eroare de tipul `MissingIpAddressException`, având următorul mesaj:

- `ADD MEMBER: MissingIpAddressException: Server IP Address was not provided. ## line no: <lineNumber>`

FIND MEMBER <params>

Comanda este utilizată pentru căutarea unui membru într-un **GRUP ȚINTĂ**. În cazul procesării cu succes, în fișierul de output vor fi scrise următoarele mesaje, în funcție de context:

- Grupul a fost identificat pe baza adresei IP, iar utilizatorul a fost regăsit în listă:
  - `FIND MEMBER: <ipAddress>: name = <name> && role = <role>`
- Grupul a fost identificat pe baza adresei IP, dar utilizatorul **nu** a fost regăsit în listă:
  - `FIND MEMBER: Member not found: ipAddress = <ipAddress>: name = <name> && role = <role>`
- Grupul nu a putut fi identificat:
  - `FIND MEMBER: Group not found: ipAddress = <ipAddress>`

Operațiunea de căutare poate genera următoarele excepții:

- Adresa IP lipsește:
  - `FIND MEMBER: MissingIpAddressException: Server IP Address was not provided. ## line no: <lineNumber>`
- Numele sau rolul utilizatorului nu au fost furnizați:
  - `FIND MEMBER: UserException: Name and role can't be empty. ## line no: <lineNumber>`

REMOVE MEMBER <params>

Comanda este folosită pentru eliminarea unui membru dintr-un **GRUP ȚINTĂ**. În cazul procesării cu succes, în fișierul de output vor fi scrise următoarele mesaje, în funcție de context:

- Grupul a fost identificat pe baza adresei IP, iar utilizatorul a fost regăsit în listă:
  - `REMOVE MEMBER: <ipAddress>: name = <name> && role = <role>`
- Grupul a fost identificat pe baza adresei IP, dar utilizatorul **nu** a fost regăsit în listă:
  - `REMOVE MEMBER: Member not found: ipAddress = <ipAddress>: name = <name> && role = <role>`
- Grupul nu a putut fi identificat:
  - `REMOVE MEMBER: Group not found: ipAddress = <ipAddress>`

Operațiunea de eliminare poate genera următoarele excepții:

- Adresa IP lipsește:
  - `REMOVE MEMBER: MissingIpAddressException: Server IP Address was not provided. ## line no: <lineNumber>`
- Numele sau rolul utilizatorului nu au fost furnizați:
  - `REMOVE MEMBER: UserException: Name and role can't be empty. ## line no: <lineNumber>`

Managementul alertelor

Fiecare alertă este legată de un server prin intermediul atributului `ipAddress`.La crearea unei noi alerte, serverul corespondent (instanța `Server` care are același `ipAddress` cu cel al alertei nou create) trebuie să notifice grupul de monitorizare (instanța `ResourceGroup` cu același `ipAddress` cu cel al alertei) pentru a le permite membrilor grupului să trateze alerta din timp.

Practic, serverele vor anunța grupurile de interes care sunt conectate la ele de apariția unei noi alerte, iar acestea o vor trata prin scrierea într-un fișier.
Evenimentele trebuie create în mod dinamic după încărcarea datelor din fișierele de input `servers_03.in`, `groups_03.in` și `events_01.in`.

ADD EVENT <params>

Comanda este folosită pentru adăugarea unui eveniment. Executarea acestei comenzi va genera următorul mesaj:

- `ADD EVENT: <ipAddress>: type = <alertType> && severity = <severityLevel> && message = <message>`

Date de intrare-ieșire

Pentru testarea funcționalităților se vor utiliza următoarele date de intrare:

- servere (`servers_xx.in`);
- grupuri de monitorizare (`groups_xx.in`);
- alerte generate de servere (`alerts_xx.in`).

Toate aceste date sunt stocate sub forma unor **fișiere CSV-like** care folosesc **caracterul "|" pe post de separator.** Compararea rezultatelor se va face prin compararea conținutului fișierelor care au extensia `.ref`, cu datele generate de voi. Datele generate vor fi stocate în fișiere cu extensia `.out`.

Exemplu: Pentru centrele de date ale căror comenzi sunt stocate în fișierul `servers_01.in`, voi veți stoca rezultatul procesării în fișierul `servers_01.out`, iar testarea presupune compararea rezultatelor din fișierul `servers_01.out` cu cele din `servers_01.ref`.

Metoda `main` din clasa `Main` poate fi invocată astfel:

- cu **doi parametri:**
  - primul va fi tipul path-ului (pe care îl veți folosi pentru a diferenția fluxul de execuție al funcționalităților);
  - al doilea este calea fișierului din care se citește, scrie și verifică rezultatul.
- cu **patru parametri:**
  - primul va fi tipul path-ului;
  - al doilea este calea către fișierele aferente serverelor;
  - al treilea este calea către fișierele aferente grupurilor de monitorizare;
  - al patrulea este calea către fișierele aferente alertelor de sistem.

Punctaj***

- 35p: testele automate de validare a rezultatelor (7p per test);
- **40p: folosirea a 4 design patterns și argumentarea alegerii lor.**
- 15p: folosirea principiilor OOP și a conceptelor învățate (moștenire, polimorfism, genericitate, error handling, etc.).
- 10p: code style (denumirea intuitivă a variabilelor și metodelor, utilizarea single-responsability principle[1], utilizarea corectă a spațierii, etc.).

Notă:** Nefolosirea a minim 4 design pattern-uri sau lipsa argumentării lor în fișierul **README.md duce la pierderea întregului punctaj.

[1] https://en.wikipedia.org/wiki/Single-responsibility_principle