Proiect POO 2025
Aplicație turistică

Responsabil: Ilie Cristian Dorobăț
Deadline: 02 februarie, ora 20.00
Assignment: https://classroom.github.com/a/CTboCaEn
Template: https://github.com/ACS-POO-2CB/proiect-2025-template -> dacă intervin corectări ale testelor de aici vă puteți lua varianta actualizată.
Versiune document: update 2

Introducere

Scopul acestui proiect îl reprezintă dezvoltarea back-end-ului unei aplicații turistice care trebuie să permită utilizatorilor atât crearea unei baze de date cu muzeele deschise publicului larg, cât și crearea grupurilor turistice.

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.

Entități de manipulare a bazei de date

Database

Clasa `Database` este definită pentru simularea interacțiunii cu baza de date. Aceasta trebuie creată utilizând un design pattern care previne instanțierea multiplă și va cuprinde următoarele atribute:

- `Database database` - instanța baze de date;
- `Set<Museum> museums` - colecția de muzee;
- `Set<Group> groups` - colecția de grupuri turistice.

și următoarele metodele:

- `addMuseum` - adaugă o singură entitate muzeală.
- `addMuseums` - adaugă o colecție întreagă de entități muzeale.
- `addGroup` - adaugă un singur grup turistic.
- `addGroups` - adaugă o colecție de grupuri turistice.

Entități pentru gestionarea muzeelor

Museum

Clasa `Museum` este definită pentru a facilita manipularea datelor care descriu **entitățile muzeale.** Aceasta cuprinde următoarele câmpuri obligatorii:

- `String name;`
- `long code;`
- `long supervisorCode;`
- `Location location;`

și următoarele câmpuri opționale:

- `Person manager;`
- `Integer foundingYear;`
- `String phoneNumber;`
- `String fax;`
- `String email;`
- `String url;`
- `String profile;`
- alte câmpuri de cate aveți nevoie pentru managementul evenimentelor (vezi pag. 6).

Location

Clasa `Location` este definită pentru a gestiona și organiza **datele asociate locului în care se află entitățile muzeale.** Aceasta cuprinde următoarele câmpuri obligatorii:

- `String county;`
- `Integer sirutaCode;`

și următoarele câmpuri opționale:

- `String locality;`
- `String adminUnit;`
- `String address;`
- `Integer latitude;`
- `Integer longitude;`

Entități pentru gestionara grupurilor turistice

Person

Clasa `Person` este utilizată pentru **definirea următoarelor tipuri de persoane:** manager-ul muzeului, vizitatori (studenți și profesori) și ghizi (profesori). Această entitate conține următoarele atribute:

- `String surname;`
- `String name;`
- `String role;`
- `int age;`
- `String email;`

Singurul constructor expus are următorii parametri: `surname, name și role`.

Student

Clasa `Student` este o **clasă specifică** prin care sunt descrise persoanele care frecventează cursurile unei unități de învățământ. Este utilizată pentru **definirea membrilor grupurilor turistice** iar pe lângă atributele și metodele **moștenite** din clasa `Person` se mai adaugă următoarele două:

- `String school;`
- `int studyYear;`

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

Professor

Clasa `Professor` este, de asemenea, o **clasă specifică** prin intermediul căreia sunt definite persoanele care predau în cadrul unei unități de învățământ. Este utilizată în primul rând la **definirea ghidului care va conduce grupul turistic,** dar și pentru **definirea altor membri ai grupului.** La fel ca în cazul clasei `Student`, și clasa `Professor` **moștenește** atributele și metodele clasei `Person`, iar pe lângă acestea mai sunt definite următoarele două:

- `int experience;`
- `String school;`

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

Group

Clasa `Group` este folosită pentru definirea grupurilor turistice. Aceasta cuprinde următoarele atribute:

- `List<Person> members` - lista de membri ai grupului;
- `Professor guide` - ghidul grupului;
- `Integer museumCode` - codul unității muzeale;
- `String timetable` - intervalul de timp pentru care grupul este programat.

și următoarele metode:

- `addGuide` - adaugă ghidul grupului;
- `addMember` - adaugă un membru în cadrul grupului;
- `removeMember` - elimină un membru din grup;
- `resetGuide` - șterge ghidul grupului.

Entități pentru gestionarea excepțiilor

Denumire	Mesaj
`GroupNotExistsException`	GroupNotExistsException: Group does not exist.
`GroupThresholdException`	GroupThresholdException: Group cannot have more than 10 members.
`GuideExistsException`	GuideExistsException: Guide already exists.
`GuideTypeException`	GuideTypeException: Guide must be a professor.
`PersonNotExistsException`	PersonNotExistsException: Person was not found in the group.

Enumerări

Clasă	Atribute

`PathTypes`	`GROUPS`
	`LISTENER`
	`MUSEUMS`

Funcționalități și comenzi

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

Adăugarea entităților muzeale

Muzeele (instanțele clasei `Museum`) trebuie create în mod dinamic după încărcarea datelor din fișierele de input `museums_xx.in`, unde `xx` reprezintă un index format din două numere (ex.: 01, 02, etc.).

ADD MUSEUM <params>

Comanda este folosită pentru adăugarea unui muzeu, fiind executată cu succes dacă nu apare o eroare.

- `<museumCode>: <museumName>`

În cazul în care este captată o eroare de tipul `IndexOutOfBoundsException`, `NullPointerException` sau `NumberFormatException`, următorul mesaj va fi înregistrat: `Exception: Data is broken. ## (<line>)`

Adăugarea grupurilor turistice

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

***Notă**: parametrul `<person>` este reprezentarea `toString` a obiectului, iar aceasta va cuprinde toate perechile proprietate-valoare separate prin virgulă după următorul format: `<prop1>=<value1>, <prop2>=<value2>, ...`*

Notă**: Pentru fiecare dintre următoarele 6 comenzi, trebuie identificat grupul turistic din întreaga colecție ale cărui valori ale atributelor `museumCode` și `timetable` corespund cu cele din fișierul de input **(vom numi această operație MAIN GROUP).

ADD GUIDE <params>

Comanda este folosită pentru adăugarea unui ghid. **Dacă MAIN GROUP există** se parcurg următoarele scenarii:

- dacă **deja există un ghid** setat pentru acel grup, trebuie aruncată o eroare cu mesajul `"GuideExistsException: Guide already exists."`
  - `<museumCode> ## <timetable> ## GuideExistsException: Guide already exists. ## (new guide: <person>)`
- dacă **grupul nu are un ghid** (un ghid poate fi șters prin comanda REMOVE GUIDE):
  - dacă este adăugat un **student pe post de ghid,** se aruncă eroarea `"GuideTypeException: Guide must be a professor."`
    - `<museumCode> ## <timetable> ## GuideTypeException: Guide must be a professor. ## (new guide: <person>)`
  - dacă **este adăugat un profesor,** atunci operația se finalizează cu succes.
    - `<museumCode> ## <timetable> ## new guide: <person>`

Dacă MAIN GROUP nu există, este creat un nou grup turistic.

FIND GUIDE <params>

Comanda permite căutarea unui ghid turistic atribuit unui grup. Dacă **persoana căutată este de tipul** `Professor`, se urmează scenariile de mai jos:

- dacă **grupul nu există,** se aruncă excepția `"GroupNotExistsException: Group does not exist.";`
  - `<museumCode> ## <timetable> GroupNotExistsException: Group does not exist. ## (find guide: <person>)`
- dacă **grupul există,** operația se finalizează cu succes.
  - `<museumCode> ## <timetable> ## guide found: <person>`
  - `<museumCode> ## <timetable> ## guide not exists: <person>`

Dacă **persoana căutată nu este de tipul** `Professor`, se aruncă excepția `"GuideTypeException: Guide must be a professor."`

- `<museumCode> ## <timetable> ## GuideTypeException: Guide must be a professor. ## (new guide: <person>)`

REMOVE GUIDE <params>

Comanda este utilizată pentru ștergerea ghidului dintr-un grup turistic. Dacă **MAIN GROUP există,** el este eliminat

- `<museumCode> ## <timetable> ## removed guide: <person>`

iar în caz contrar este aruncată o eroare cu mesajul `"GroupNotExistsException: Group does not exist."`

- `<museumCode> ## <timetable> ## GroupNotExistsException: Group does not exist. ## (removed member: <person>)`

ADD MEMBER <params>

Comanda permite adăugarea unui membru în grup (membrii grupului pot fi atât studenți cât și profesori). **Dacă MAIN GROUP există,** se parcurg scenariile următoare:

- dacă **grupul are mai puțin de 10 membri,** membrul nou este adăugat;
  - `<museumCode> ## <timetable> ## new member: <person>`
- dacă **grupul are deja 10 membri,** este aruncată eroarea `"GroupThresholdException: Group cannot have more than 10 members."`
  - `<museumCode> ## <timetable> ## GroupThresholdException: Group cannot have more than 10 members. ## (new member: <person>)`

Dacă MAIN GROUP nu există, atunci este aruncată eroarea `"GroupNotExistsException: Group does not exist."`

- `<museumCode> ## <timetable> ## GroupNotExistsException: Group does not exist. ## (new member: <person>)`

FIND MEMBER <params>

Comanda este utilizată pentru căutarea unui membru dintr-un grup turistic anume. Dacă nu a fost identificat grupul, atunci se aruncă eroarea `"GroupNotExistsException: Group does not exist."`,

- `<museumCode> ## <timetable> ## GroupNotExistsException: Group does not exist. ## (find member: <person>)`

altfel, operația se finalizează cu succes.

- `<museumCode> ## <timetable> member found: <person>`
- `<museumCode> ## <timetable> member not exists: <person>`

REMOVE MEMBER <params>

Comanda este folosită pentru eliminarea unui membru din grupul turistic. **Dacă MAIN GROUP există,** se urmează scenariile de mai jos:

- dacă **membrul există în lista** de membri a grupului, el este eliminat;
  - `<museumCode> ## <timetable> removed member: <person>`
- dacă membrul nu există în grup, este aruncată eroarea `"PersonNotExistsException: Person was not found in the group."`
  - `<museumCode> ## <timetable> ## PersonNotExistsException: Person was not found in the group. ## (<person>)`

Dacă MAIN GROUP nu există, este aruncată eroarea `"GroupNotExistsException: Group does not exist."`

- `<museumCode> ## <timetable> ## GroupNotExistsException: Group does not exist. ## (removed member: <person>)`

Adăugarea evenimentelor

Evenimentele nu sunt entități corespondente unei clase, ci sunt notificări transmise de entitățile muzeale (instanțele clasei `Museum`) către ghizii turistici înregistrați (atributul `guide`

al instanțelor clasei `Group`). De fiecare dată când un muzeu anunță un eveniment, acesta trebuie transmis către toți ghizii înregistrați cu un grup de vizită al acelui muzeu pentru a le aduce la cunoștință de existența unor modificări în ceea ce privește orarul de funcționare ori de organizarea altor expoziții.

Evenimentele trebuie create în mod dinamic după încărcarea datelor din fișierele de input `events_01.in`.

ADD EVENT <params>

Comanda este folosită pentru adăugarea unui eveniment. Executarea acestei comenzi va genera un mesaj sub forma `To: <emailAddress> ## Message: <museumName> (<museumCode>) <organizerMessage>`

Date de intrare-iesire

Pentru testarea funcționalităților se vor utiliza următoarele date de intrare:

- entități muzeale din România (`museums_xx.in`);
- grupuri turistice (`groups_xx.in`);
- evenimente generate de entitățile muzeale (`events_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 cu datele stocate de voi în fișiere cu extensia `.out` față de fișierele cu extensia `.ref`.

Exemplu: Pentru entitățile muzeale ale căror comenzi sunt stocate în fișierul `museums_01.in`, voi veți stoca rezultatul procesării în `museums_01.out`, iar testarea presupune compararea rezultatelor din fișierul `museums_01.out` cu cele din `museums_01.ref`.

Metoda main din clasa Main poate primi **2 sau 4 parametri.**

- în cazul în care metoda primește **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.
- dacă metoda primește **patru parametri:**
  - primul va fi tipul path-ului;
  - al doilea este calea către fișierele aferente muzeelor;
  - al treilea este calea către fișierele aferente grupurilor turistice;
  - al patrulea este calea către fișierele aferente evenimentelor.

Punctaj

- 35p: testele automate de validare a rezultatelor (5p 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:** Punctajul este acordat doar dacă pentru validarea testelor sunt **utilizate cel puțin 4 design pattern-uri** parcurse la curs iar alegerea acestora este argumentată în fișierul **README.md. În caz contrar, se pierde tot punctajul.***

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