B. Zboruri

Cerinta: Construiti un sistem cu functionalitati reduse care sa ajute la gestionarea mai usoara a zborurilor. Aplicatia pe care o veti crea va avea posibilitatea de a crea, cauta si modifica zboruri. Pentru o mai buna gestionare a codului a fost creata o structura de clase de la care veti porni cu implementarea. Asadar, urmarind diagramele UML de mai jos, dar si ceritele specifice de pe urmatoarea pagina, dezvoltati aceasta aplicatie.

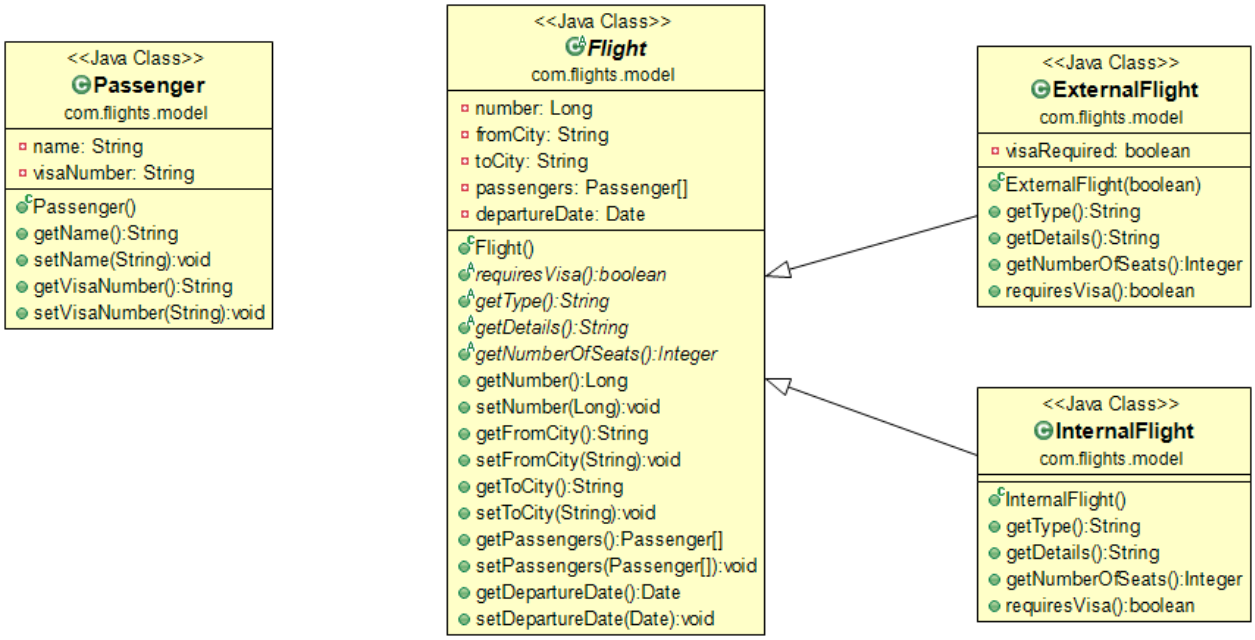

Figure 1: Diagrama UML pentru modele

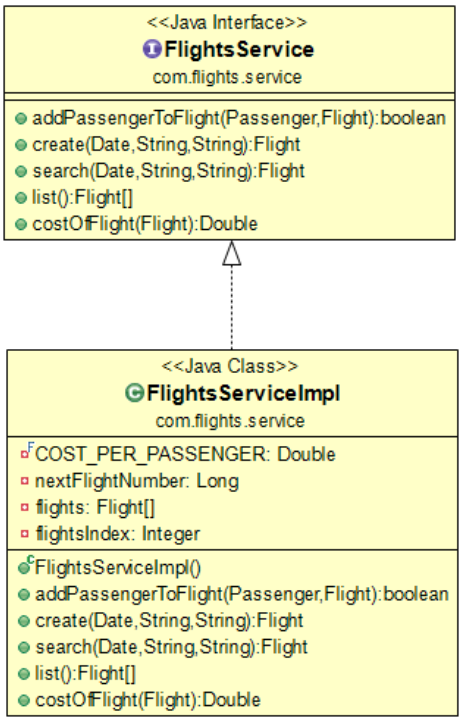

Figure 2: Diagrama UML pentru servicii

1. Cititi si intelegeti cerinta si diagramele UML. Descarcati si incarcati in IDE scheletul aplicatiei.
2. Creati clasele `InternalFlight` si `ExternalFlight` ca si subclase ale clasei `Flight`. Implementati metodele abstracte avand in vedere urmatoarele puncte:

- tipul unui `InternalFlight` trebuie sa fie sirul de caractere "INTERNAL", respectiv "EXTERNAL" pentru celalalt tip de zbor
- in cazul unui zbor extern numarul de locuri este 200, iar pentru zborul intern 100
- mesajul pe care trebuie sa-l intoarca metoda `getDetails` trebuie sa aiba forma *"Zbor nr.: <nr.-zbor>, data plecarii: <data-plecarii>, orasul de plecare: <orasul-de-plecare>, destinatia: <destinatia>, tip: <INTERNAL/EXTERNAL>, necesita viza: <DA/NU>"*. In mesaj afisati informatia despre viza doar in cazul zborurilor externe.

3. Implementati functionalitatea pentru crearea unui zbor si pentru adaugarea unui nou pasager la un anume zbor dupa cum urmeaza:

- la creare al doilea parametru reprezinta orasul de plecare, iar cel de-al treilea parametru reprezinta orasul destinatie. Folositi clasa `DateService` pentru a crea o instanta a clasei `Date`.
- cand creati un zbor asigurati-va ca datele pe care le primiti sunt valide, adica non-null. De asemenea, un zbor este considerat ca fiind intern daca orasele de plecare si destinatie incep cu aceeasi litera. Mai mult, daca orasele au acelasi numar de caractere este nevoie de viza pentru acel zbor; asigurati-va ca informatia despre viza se poate salva o singura data, la crearea obiectului, si ca nu poate fi modificata ulterior.
- cand veti incerca sa adaugati un pasager la un anume zbor sa va asigurati, dinnou, ca datele sunt valide. De asemenea, sa luati in calcul, pentru zborul respectiv, daca pasagerul are nevoie de viza sau nu.

4. Implementati `search` si `list` dupa cum urmeaza:

- metoda de cautare trebuie sa intoarca primul rezultat pe care-l gaseste pentru data si orasele respective. Daca zborul nu exista se va intoarce **`null`**
- metoda de listare va afisa detaliile tuturor zborurilor diferite de **`null`** ordonate crescator dupa numarul zborului dupa care va intoarce lista pe care tocmai ati afisat-o.

5. Implementati metoda `costOfFlight` care sa calculeze costul unui zbor in functie de numarul de pasageri diferiti de **`null`** al zborului respectiv; luati in calcul numarul de pasageri. nu numarul de locuri. Folositi constanta `COST_PER_PASSENGER` la care-i puteti da orice valoare.

6. Scrieti niste teste in clasa `com.flights.web.FlightsTest` in care sa demonstati functionalitatea corecta a implementarii serviciului `FlightsService`.

Note: In order to add JUnit to your project do the following: Right click on the project → Build Path → Add libraries... → JUnit