Univerzitet u Sarajevu
Elektrotehnički fakultet Sarajevo
Predmet: Razvoj programskih rješenja / Razvoj softvera 2019/2020

# II parcijalni ispit, septembarski rok

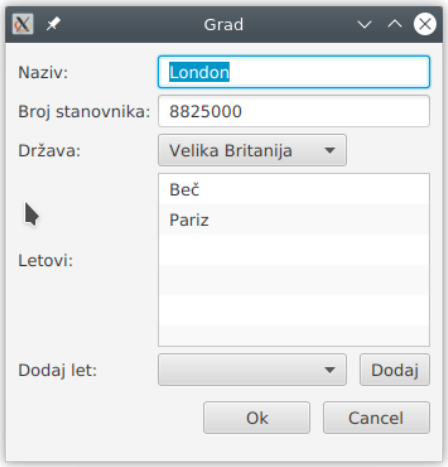
***Pozivnica za GitHub Classroom:*** https://classroom.github.com/a/QidSZ1Ed

***Ukupno bodova: 20*** (17 bodova će se dodijeliti proporcionalno broju uspješnih testova, pri čemu se početnih 15 testova ne računaju, ali će se oduzimati bodovi ako ti testovi budu padali. Ostalih 3 boda će se dodijeliti ručnim pregledanjem.) Studenti predmeta Razvoj softvera rade i zadatak 2 koji nosi dodatnih 10 bodova (5 na osnovu testova a 5 ručnim pregledanjem).

Na repozitoriju se nalazi gotov projekat koji predstavlja tutorijal 9 sa Java FX korisničkim interfejsom. Projekat sadrži i 15 testova koji rade (služe za provjeru regresije), te 17 testova u klasama koje se zovu Ispit* koje su zakomentarisane, a koji se tiču zadatka 1 navedenog ispod. Kada uradite zadatak trebate odkomentarisati klase kako bi testovi prošli. Na poleđini ovog ispita imate detaljniji opis programa koji bi vam mogao pomoći da se snađete u kodu.

**Zadatak 1:**

Na formular za dodavanje i izmjenu grada dodajte ListView **lvLetovi** koji označava za koje sve gradove postoji direktan let avionom između odabranog grada i tog grada. Ispod lvLetovi treba se nalaziti padajuća lista **choiceGrad** sa spiskom svih gradova i dugme **btnDodajLet** koje omogućuje dodavanje novog leta. Prozor za editovanje grada treba izgledati kao na slici:




Obratite pažnju da ako dodate let npr. Pariz-London time ste automatski dodali i let u obrnutom pravcu London-Pariz, također i da ne možete dvaput dodati isti let.

Također dodajte sva potrebna polja u bazu podataka (datoteku baza.db.sql), te potrebne metode i

atribute u sve klase kako bi podaci ostali trajno sačuvani. Zadatak obavezno realizirajte tako što ćete u klasu Grad dodati atribut **letovi** tipa **TreeSet<Grad>** te odgovarajući setter i getter setLetovi i getLetovi. Klasa GradController treba imati dva konstruktora: pored ovog koji je dat (sa dva parametra), treba napraviti i konstruktor koji ima treći parametar tipa ArrayList<Grad> kako bi se popunila padajuća lista choiceGrad.

Pored 17 testova u klasama Ispit* biće ručnim pregledanjem dodijeljena 3 boda ako prozor grad.fxml izgleda identično kao na slici iznad, te ako slijedi pravila dizajna naučena na predmetu.

*Savjet: Ako ste dodali novu tabelu u bazu, obavezno trebate ažurirati i metodu u DAO klasi* ***vratiBazuNaDefault****.*

**Zadatak 2 (samo predmet Razvoj softvera):**

Na glavnu formu dodajte dugmad "Zapiši JSON" (**btnZapisiJSON**) i "Učitaj JSON" (**btnUcitajJSON**). Ova dugmad trebaju otvoriti standardne open/save dijaloge za odabir datoteke, a zatim zapisati odnosno učitati JSON datoteku u formatu koji je opisan na sljedećoj stranici. U slučaju neispravno formatirane datoteke, dugme "Učitaj JSON" treba prikazati Alert tipa ERROR.

Za potrebe realizacije ovog zadatka kreirajte klasu **JSONFormat** koja ima: privatne atribute **ArrayList<Drzava> drzave** i **ArrayList<Grad> gradovi** sa setterima i getterima, metodu **ucitaj** koja prima **File** i iz datoteke puni podatke u privatne atribute, te metodu **zapisi** koja prima **File** i u datoteku upisuje podatke iz privatnih atributa. Metoda ucitaj treba baciti izuzetak za sve vrste neispravnih JSON datoteka, uključujući situacije kada neko od polja označenih kao **obavezno** ne postoji u datoteci.

Zadatak 3 nosi 10 bodova, od čega će 5 biti dodijeljeno pomoću testova u klasi RSIspitJSONTest, a 5 ručnim pregledanjem glavnog prozora, tražene dugmadi i pratećih akcija.

Sarajevo, 12. 9. 2020

**Opis projekta**

SQLite baza podataka **baza.db** (nalazi se u korijenskom direktoriju projekta) sadrži dvije tabele:

- Tabela grad sadrži kolone: id (int, primarni ključ), naziv (text), broj_stanovnika (int), drzava (int, strani ključ)
- Tabela drzava sadrži kolone: id (int, primarni ključ), naziv (text), glavni_grad (int, strani ključ)

Dump baze nalazi se u datoteci **baza.db.sql**.

Na osnovu ove dvije tabele formirane su DTO klase **Grad** i **Drzava** koje se pridržavaju JavaBean specifikacije, sadrže sve pobrojane atribute, gettere, settere, konstruktor bez parametara i konstruktor sa svim atributima kao parametrima. Atribut drzava u klasi Grad je tipa Drzava, a atribut glavniGrad u klasi Drzava je tipa Grad (reference na klase).

Klasa **GeografijaDAO** predstavlja tipičnu DAO klasu, ova klasa je singleton što znači da je konstruktor privatan, a metodom getInstance se dobija referenca na ovu klasu. getInstance poziva konstruktor. Konstruktor klase najprije pokušava izvršiti jedan upit da ustanovi da li datoteka baza.db postoji. Ako ne postoji, poziva se metode regerenisiBazu() koja kreira novu bazu iz dump datoteke, izvršavajući upite koji se u njoj nalaze. Zatim se u konstruktoru kreiraju svi pripremljeni upiti koji su potrebni za ostatak klase.

Pored ovih, klasa GeografijaDAO sadrži metode iz tutorijala 9:

- Grad glavniGrad(String nazivDrzave) - vraća null ako država ne postoji
- void obrisiDrzavu(String nazivDrzave) - briše i sve gradove u toj državi
- ArrayList<Grad> gradovi() - vraća spisak gradova sortiranih po broju stanovnika u opadajućem redoslijedu
- void dodajGrad(Grad grad) i void dodajDrzavu(Drzava drzava)
- void izmijeniGrad(Grad grad) - ažurira slog u bazi za dati grad
- Drzava nadjiDrzavu(String nazivDrzave) - vraća null ako država ne postoji

Kao i sljedeće metode kojih nema u tutorijalu 9:

- Grad nadjiGrad(String nazivGrada)
- void obrisiGrad(Grad grad)
- ArrayList<Drzava> drzave()
- void vratiBazuNaDefault() - služi da bi se baza podataka vratila na stanje koje je dato u datotei baza.db.sql (zbog testova koji mijenjaju sadržaj baze).

U folderu **resources/fxml** nalaze se sljedeći prozori:

- **glavna.fxml** sadrži TableView (tableViewGradovi) sa spiskom gradova, te četiri dugmeta za dodavanje, promjenu, brisanje grada i dodavanje države:
    - GlavnaController je kontroler za ovu formu. Ona popunjava tableViewGradovi, te sadrži action* evente za klik na četiri dugmeta.
    - actionDodajGrad, actionIzmijeniGrad i actionDodajDrzavu otvaraju forme grad.fxml i drzava.fxml. Kod zatvaranja forme izvršava se lambda (setOnHiding) koja poziva metodu getGrad / getDrzava kontrolera te poziva metodu dodajGrad / izmijeniGrad / dodajDrzavu u DAO klasi (ako nije vraćen null).
    - actionObrisiGrad prikazuje Alert tipa Confirmation, te ako je korisnik kliknuo na OK poziva se metoda obrisiGrad iz DAO klase.

- **grad.fxml** i **drzava.fxml** su forme za unos novog grada i države / promjenu postojećih, sadrže polja koja odgovaraju atributima klasa Grad i Drzava, te dva dugmeta Ok i Cancel
    - GradController i DrzavaController imaju konstruktor sa dva parametra, prvi je Grad ili Drzava a drugi je GeografijaDAO. Ako je prvi parametar null, onda je u pitanju dodavanje novog, a ako nije onda je izmjena. DAO služi isključivo kako bi se padajuća lista choiceGrad/choiceDrzava popunila svim gradovima/državama iz baze, što se obavlja u metodi initialize(). Ova metoda također popunjava formu u slučaju izmjene.
    - Metoda clickCancel zatvara prozor i postavlja atribut grad na null.
    - clickOk vrši validaciju forme, dodaje CSS klasu poljeIspravno ili poljeNeispravno, te ako su sva polja ispravna zatvara formu. Ova metoda *ne dodaje* ništa u bazu. Umjesto toga, ona ažurira privatni atribut grad / drzava. GlavnaController će pozvati metodu getGrad / getDrzava kako bi dobili vrijednost tog privatnog atributa. U slučaju cancel privatni atribut će biti postavljen na null.