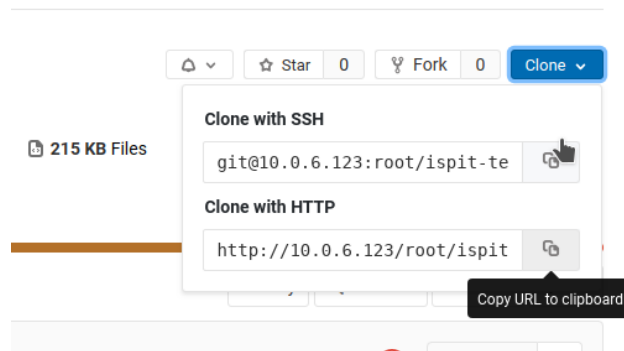


## Parcijalni ispit - Upute za rad

1. Upalite okruženje IntelliJ IDEA.
2. Pristupite web baziranom okruženju GitLab na adresi:  
<http://10.0.6.123>
3. Kliknite na opciju **Register** kako biste kreirali novi korisnički račun.
4. Unesite vaše *pravo ime i prezime*, vašu *pravu ETF email adresu*, za korisničko ime uzmite *prvi dio email* adrese prije znaka @ (npr. mbajraktar1), izaberite neki password koji nije lako pogoditi. **Jako je važno da baš ovakve podatke unesete!** U suprotnom nećete dobiti zadatak.
5. Refreshujte GitLab stranicu: ubrzo ćete ugledati repozitorij koji se zove **Ispit – test**. Otvorite taj repozitorij i kliknite na opciju **Fork**.
6. Sada se vratite na naslovnu stranicu, vidjećete dva repozitorija: jedan ste već vidjeli (Administrator / Ispit - test), a drugi je vaša kopija/fork (Vaše Ime / Ispit - test). Uđite u taj drugi repozitorij.
7. Kliknite na opciju **Clone** a zatim izaberite **Clone with HTTP**. Ovaj URL možete iskoristiti u IDEA da preuzmete projekat sa Git servera kako smo radili na tutorijalima.



8. Pokrenite projekat kako biste se uvjerali da sve radi. Trebali biste ugledati prozor "Sve je spremno za ispit".
9. Vratite se na početnu GitLab stranicu i refreshajte je. Tačno 15 minuta nakon početka ispita trebali biste ugledati novi repozitorij pod imenom **RPR I parcijalni 7.2.2019**.
10. Ponovite sve korake kao i za repozitorij Ispit – test: Fork, vratite se na početnu, otvorite vaš primjerak repozitorija, Clone, otvorite iz IDEA!
11. Radite samostalno projekat i nemojte zaboraviti raditi commit i push! Biće pregledana posljednja verzija koju push-ate.
12. Nemojte gledati ekran od kolega/kolegica pored vas i raditi isto! Zabilježili smo ko je gdje sjedio, poslije ispita biće korišten poznati sistem za provjeru prepisivanja i bićete bodovani sa 0 bodova ako se vaš kod bude neznatno razlikovao od osoba pored vas.

## I parcijalni ispit, varijanta D

**Ukupno bodova: 20** (Bodovi će se dodijeliti proporcionalno broju uspješnih testova.)

Na repozitoriju se nalazi gotov projekat koji sadrži samo praznu Main klasu i testove. Vaš zadatak je da napravite kompletan Java program koji zadovoljava postavku zadatka i prolazi testove.

Svojim potpisom student izjavljuje da je saglasan sa ovim sistemom bodovanja i da se rješenje postavljenog zadatka nalazi na serveru kako je objašnjeno u uputama za izradu ispita.

### Zadatak:

Potrebno je implementirati Java projekat koji treba da sadrži sljedeće klase, sa navedenim atributima i metodama. Pored tih vi možete dodati i druge klase, metode i attribute po želji kako biste ispunili zadatak. Pri tome se pridržavati pravila vezanih za pisanje kvalitetnog koda obrađenih na predmetu, između ostalog svi atributi obavezno moraju biti privatni.

1. Klasu ili interfejs **Osoba** iz koje su izvedene klase ili interfejsi **Student** i **Nastavnik**. Iz **Student** su izvedene klase **BachelorStudent** i **MasterStudent**, a iz **Nastavnik** klase **Docent**, **VanredniProfesor** i **RedovniProfesor**.

2. Sve nabrojane klase sadrže (nasljeđuju) atribut **imePrezime** (jedan string), standardne settere i gettere za ovaj atribut te konstruktor koji prima string imePrezime.

3. Sve klase se trebaju moći ispisati na ekran koristeći System.out.println, npr. ovako:  
`Student s = new BachelorStudent("Sara Sarac");`  
`System.out.println(s);`

4. Ispis na ekran u slučaju odgovarajućih klasa treba izgledati ovako:

Za BachelorStudent: "Bachelor student Sara Sarac (12345), prosjek ocjena: 8.7"

Za MasterStudent: "Master student Sara Sarac (12345), prosjek ocjena: 8.7"

Za Docent: "Doc. dr Sara Sarac"

Za VanredniProfesor: "V. prof. dr Sara Sarac"

Za RedovniProfesor: "R. prof. dr Sara Sarac"

Prosjek ocjena mora biti zaokružen na jednu decimalu.

5. Broj 12345 u zagradi je broj indeksa. Sve klase izvedene iz Student trebaju imati privatni atribut **brojIndeksa** tipa String, te odgovarajući getter i setter. Default vrijednost indeksa je prazan string "".

6. Klasa Student treba imati privatni atribut **ocjene** koji je dat kao običan niz (a ne lista ili nešto slično!) tipa **int** sa fiksno 100 elemenata. Kod rješenja koja budu koristila nešto što nije niz, odgovarajući testovi neće biti priznati.

7. Također treba imati metodu **dodajOcjenu** tipa void koja dodaje jednu ocjenu u niz ocjena. Ako ocjena nije na intervalu [5,10] treba baciti izuzetak tipa **IlegalnaOcjena** sa tekstom poruke "Ilegalna ocjena 3" (gdje se umjesto broja 3 treba nalaziti ocjena koja je poslana metodi). U slučaju da se

prekorači kapacitet niza treba baciti izuzetak tipa **IllegalArgumentException** sa tekstom poruke "Dosegnut maksimalan broj ocjena".

8. Konačno ove klase trebaju imati i metodu **prosjek** koja vraća vrijednost tipa **double** koja predstavlja srednju vrijednost dodatih ocjena, ili broj 0 ako nije dodata niti jedna ocjena.

9. U klasi **MasterStudent** treba se nalaziti drugi niz koji predstavlja ocjene na drugom ciklusu studija, kao i metode **dodajOcjenuMaster** koja dodaje ocjenu u taj niz, i **prosjekMaster** koja vraća prosjek ocjena na drugom ciklusu. Metoda **prosjek** klase **MasterStudent** treba vraćati prosjek ocjena za *oba ciklusa zajedno* (posmatrajući ta dva niza kao jedan).

10. Program treba sadržavati i klasu **Fakultet** koja kao privatni atribut ima neku kolekciju osoba. Treba posjedovati metodu **dodajOsobu** koja stavlja novu osobu u tu kolekciju. Ako se fakultet proba ispisati na ekran metodom `System.out.println`, treba ispisati sve osobe koje su dodane u fakultet, svaku u zasebnom redu, onim redom kojim su dodavane.

11. Klasa **Fakultet** treba imati metodu **studenti** koja vraća skup (**Set**) studenata koji sadrži sve studente na fakultetu, sortirane po prosječnoj ocjeni od veće ka manjoj. Skup treba biti tipa **Student** a ne tipa **Osoba**!

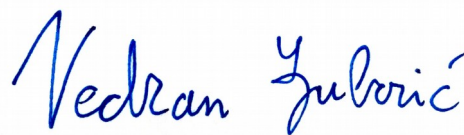
12. Dalje, treba imati metodu **filtriraj** koja prima *lambda funkciju* koja prima osobu a vraća boolean vrijednost. Metoda **filtriraj** vraća listu (**List**) svih osoba za koje je lambda funkcija vratila **true**.

13. Treba imati i metodu **topBachelor** koja vraća listu svih BachelorStudenata koji imaju prosjek veći ili jednak 8. Metoda treba vraćati listu **BachelorStudenata**, a ne listu **Osoba**! Treba imati *dvije linije koda*, od čega je jedna poziv metode **filtriraj**. Za metode implementirane na neki drugi način neće biti priznati odgovarajući testovi.

14. Klase **MasterStudent** i **Docent** (ali ne i ostale klase!) trebaju imati metodu **rodjendan** koja prima **datumRodjenja** (`LocalDate`) a vraća string "Sretan rođendan!" ako na datum izvršavanja programa pada rođendan osobe, u suprotnom vraća prazan string "".

15. Pored toga klasa **Fakultet** treba imati metodu **mladi** koja vraća **List<Mladi>** koja sadrži samo osobe koje ispunjavaju prethodni uslov.

Sarajevo, 18. 2. 2019

A handwritten signature in blue ink that reads "Vedran Zuborić". The signature is written in a cursive, flowing style.

## II parcijalni ispit, varijanta B

**Ukupno bodova: 20** (Bodovi 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.)

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 15 testova u klasama koje se zovu Ispit\* koje su zakomentarisane, a koji se tiču zadatka 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:

Potrebno je u tabelu **grad** dodati kolonu zagađenost tipa integer.

*Napomena: Mada je na računarima instaliran SQLite Browser, ne morate ga koristiti. Dovoljno je (i potrebno) da editujete fajl baza.db.sql i da u definiciju tabele **grad** dodate kolonu:*

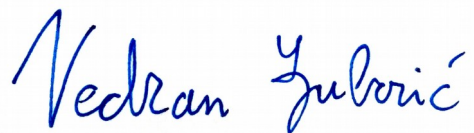
*`zagađenost' INTEGER,*

*Također u INSERT direktive dodajte još jedan cijeli broj na kraj koji možete izmisliti, nije bitan i neće se testirati.*

Ovo polje ima validnu vrijednost između 1 (zrak nije zagađen) i 10 (opasno po život). U slučaju da se odgovarajućim metodama klase Grad proba poslati različita vrijednost, treba baciti izuzetak tipa IllegalArgumentException. Na grafičkom okruženju zagađenost treba biti predstavljena kao kontrola tipa Slider.

Pored toga, na glavnoj formi (ispod postojećih dugmadi) treba se nalaziti dugme Zapiši (fx:id btnZapisi). Klikom na ovo dugme treba se izvršiti serijalizacija podataka iz baze u XML datoteku tako da osnovna klasa bude klasa Geografija koja sadrži listu svih gradova i država.

Sarajevo, 18. 2. 2019



## 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 attribute, 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 privatn, 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 regenerisiBazu() 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
- void removeInstance() - služi da bi se prekinula konekcija na bazu, kako bi se fajl baza.db mogao obrisati (što se koristi u testovima).

Kao i sljedeće metode kojih nema u tutorijalu 9:

- Grad nadjiGrad(String nazivGrada)
- void obrisiGrad(Grad grad)
- ArrayList<Drzava> drzave()

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.
  - Metoda resetujBazu() se poziva iz testova kako bi baza podataka bila regenerisana.
- **grad.fxml** i **drzava.fxml** su forme za unos novog grada i države / promjenu postojećih, sadrže polja koja odgovaraju atributima klase 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.