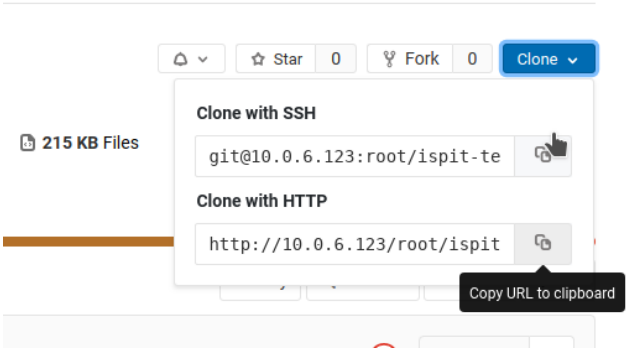
Elektrotehnički fakultet Sarajevo
Predmet: Razvoj programskih rješenja
Akademska 2018/2019. godina

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 uvjerili 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 repositorij 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.

Sarajevo, 7. 2. 2019

Vedran Zubrić

Elektrotehnički fakultet Sarajevo
Predmet: Razvoj programskih rješenja
Akademska 2018/2019. godina

Potpis studenta: _____

I parcijalni ispit, varijanta C (10:00)

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 atribute 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.

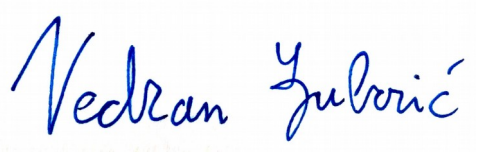
1. Klasa **Radnik** treba da sadrži:
- Privatne atribute **imePrezime** i **jmbg** tipa string, njihove settere i gettere, te konstruktor koji prima imePrezime i jmbg kao parametre. Klasa ne smije imati konstruktor bez parametara.
- Privatni atribut **plate** (običan niz tipa double sa fiksno 1000 elemenata).
- Metodu **dodajPlatu** koja primljenu vrijednost tipa double dodaje u spomenuti niz. Ako se dostigne kapacitet niza, treba baciti izuzetak tipa IllegalArgumentException sa tekstom "Ne možete registrovati više od 1000 plata za radnika Pero Perić" (gdje umjesto Pero Perić treba stajati stvarno ime i prezime radnika).
- Metodu **prosjecnaPlata** koja vraća srednju vrijednost svih registrovanih plata, ili vrijednost 0 ako nije registrovana niti jedna plata za ovog radnika.

2. Klasa **RadnoMjesto** treba da sadrži:
- Privatne atribute: **naziv** (string), **koeficijent** (double) i **radnik** (referenca na klasu Radnik koja može biti null, što znači da na spomenuto radno mjesto nije primljen niti jedan radnik).
- Klasa treba slijediti JavaBean specifikaciju.

3. Klasa **Preduzece** treba da sadrži:
- Privatni atribut **osnovica** (prirodan broj), metode **dajOsnovicu** i **postaviOsnovicu** koje predstavljaju getter i setter za taj atribut, te konstruktor sa jednim parametrom koji također postavlja atribut.
- Ako se pokuša postaviti negativna vrijednost osnovice ili nula, metoda postaviOsnovicu i konstruktor trebaju baciti izuzetak tipa **NeispravnaOsnovica** čiji je tekst poruke "Neispravna osnovica -123" gdje umjesto broja -123 treba stajati vrijednost osnovice koja je proslijeđena. Naravno, u tom slučaju vrijednost osnovice treba ostati neizmijenjena.

- Metodu **novoRadnoMjesto** koja prima parametar tipa RadnoMjesto i dodaje ga u neku internu kolekciju koja sadrži sva radna mjesta u firmi. Obratite pažnju da se u jednoj firmi može nalaziti više identičnih radnih mjesta, npr. neka firma može imati 10 vozača, pa u tom slučaju možete dodati 10 radnih mjesta koja sva imaju naziv "vozač" i isti koeficijent, ali na njima se nalaze različiti radnici.
- Metodu **zaposli** koja prima dva parametra: Radnik i naziv radnog mjesta (string). Ova metoda treba u internoj kolekciji potražiti da li se nalazi radno mjesto pod tim imenom i da li je ono popunjeno. Ako radno mjesto ne postoji ili su sva takva mjesta popunjena, treba baciti izuzetak tipa **IllegalStateException** sa tekstom poruke "Nijedno radno mjesto tog tipa nije slobodno". U suprotnom, treba pridružiti radnika na prvo slobodno radno mjesto sa tim nazivom.
- Metodu **obracunajPlatu** koja za sve radnike koji su evidentirani na radnim mjestima u firmi izračunava platu po formuli **osnovica*koeficijent**, te dodaje platu radniku pozivajući metodu dodajPlatu klase Radnik. Ova metoda je tipa void i nema parametara.
- Metoda **iznosPlate** je slična kao prethodna, ali ona ne treba dodati plate radnicima nego vratiti double vrijednost koja predstavlja iznos koliko bi ukupno trebalo novca da se isplate plate svim radnicima. Dakle ova metoda ne smije promijeniti stanje.
- Metodu **radnici** koja vraća skup (**Set**) svih radnika trenutno zaposlenih u firmi. Skup treba biti sortiran po prosječnoj plati, od najmanje ka većoj.
- Metodu **sistematizacija** koja vraća mapu (**Map**) u kojoj je ključ RadnoMjesto, a vrijednost je cijeli broj koji označava koliko takvih radnih mjesta ima u firmi. Recimo u nekoj programerskoj firmi postoji 100 radnih mjesta koja se sva zovu "programer" i imaju isti koeficijent, a nema drugih radnih mjesta. U tom slučaju ova metoda bi trebala vratiti mapu sa samo jednim elementom, čiji je ključ radno mjesto sa nazivom "programer" i datim koeficijentom, a vrijednost je broj 100.
- Metodu **filterRadnici** koja prima *lambda funkciju*, vraća listu (**List**) svih radnika za koje data lambda funkcija vraća boolean vrijednost true. Dakle lambda funkcija treba primati kao parametar Radnika a vraćati boolean vrijednost.
- Metodu **vecaProsjecnaPlata** koja prima vrijednost tipa double, a vraća listu (**List**) radnika čija je prosječna plata veća od primljene vrijednosti. Ova metoda mora imati samo jednu liniju koda koja predstavlja poziv metode *filterRadnici*. Bilo kakva različita implementacija od ove neće biti priznata (neće se priznavati odgovarajući test/ovi).

Sarajevo, 7. 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 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
- 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 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.