

*Jesenji semestar, 2021/22*

*PREDMET: SE321* *OBEZBEĐENJE KVALITETA, TESTIRANJE I ODRŽAVANJE SOFTVERA*

Domaći zadatak: **08**

Ime i prezime: **Bogdan Janković**

Broj indeksa: **3920**

Datum izrade: **26.11.2021.**

**Tekst domaćeg zadatka:** Za odabranu aplikaciju u okviru prethodnih domaćih zadataka primeniti sledeće:

1. Izvršiti mocking testiranje

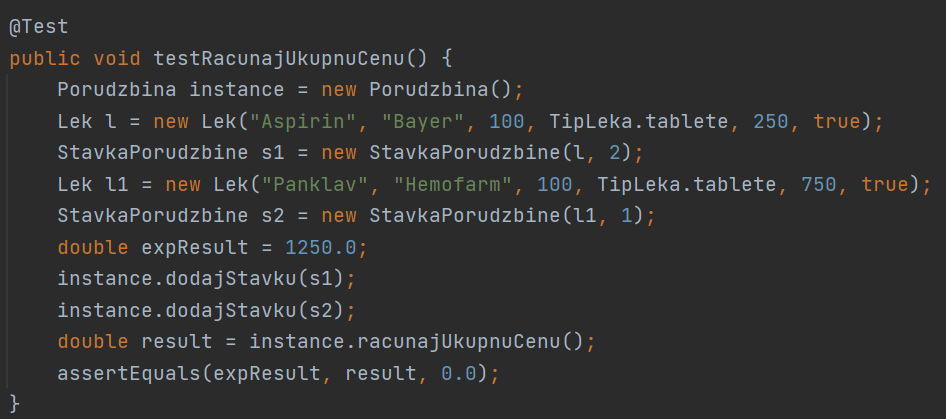
2. Izvršiti integraciono testiranje

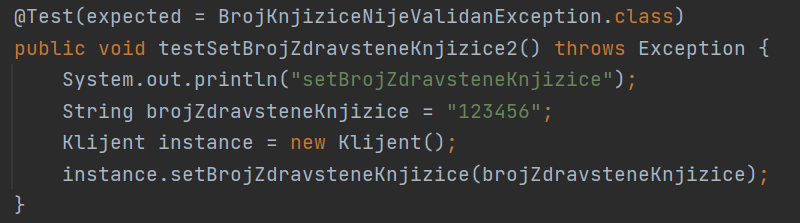
**Mocking testiranje**

Mocking pokušava da na lak način reši kreiranje lažnih objekata (klasa) koji pomažu procesu jediničnog testiranja. Tako lažni objekti su dizajnirani da „zavaraju“ Java objekt (koji se jedinično testira) tako da se ima utisak da on komunicira sa drugim stvarnim objektima.

Primeri testova sa kreiranje lažnih objekata radi samog testiranja aplikacije koja je obradjivana u okviru prethodnih domaćih zadataka:







Lažni objekti omogućavaju da postavite kontrolisano okruženje sa strogo definisanim determinističkim ponašanjem svih objekata koji su uključeni u objekat koja se testira.

**Integraciono testiranje**

Integraciono testiranje se radi posle jediničnog testiranja ( “Unit testing” ). Nakon što su sve individualne jedinice (metode) napravljene i testirane, počinjemo sa kombinovanjem ovih jedinično-iztestiranih modula i tako radimo integraciono testiranje. Integraciono testiranje se obavlja u toku razvoja softvera a ne na kraju razvoja softvera. Koristi se mogu dobiti od ovakve pažljive integacije su lakša dijagnoza defekata, manje defekata, manje vremena do prvog operativnog proizvoda, kraće ukupno vreme razvoja proizvoda,veće šanse da se projekat uradi uspešno i na samom kraju pouzdanije procene rokova i naravno bolji kvalitet programa.

Postupak integracionog testiranja:

1. individualni moduli se prvo testiraju u izolaciji od ostalih modula odnosno jedinično testiraju
2. integrisati tj. kombinovati jedinično-iztestirane module, jedan po jedan, u složenije celine
3. testirati sve dok se ne integrišu svi moduli, i to uključujući kombinovano ponašanje modula
4. testirati svaki ovaj složeni modul kao jednu jedinicu
5. glavni cilj da se testiraju interfejsi između jedinica/modula i da se izvrši validacija da li su zahtevi implementirani korektno

Koraci koji su potrebni kod integracionog testiranja, tj. izvršavanje integracionih testova:

• proučiti arhitekturu aplikacije

Apoteka poseduje bazu koja cuva sve podatke o lekovima odnosno stavkama apoteke, klijentima i prodaji. Njom se upravlja posebnim softverskim upravljackim sistemom za transakcionu obradu sto zajedno **predstavlja arhitekturu sa skladistem podataka** gde ova baza predstavlja centralni repozitorijum.

• identifikovati module

Dva osnovna modula koja čine ovaj sistem su modul farmaceuta i modul korisnika koji shodno svom načinu pristupa sistemu imaju i drugačije funkcionalnosti.

• identifikovati šta svaki modul radi

Modul Farmaceut: omogućava autentifikaciju farmaceuta koji pristupa sistemu i to unošenjem jedinstvenog pin-a, zatim pregled svih lekova dostupnih u apoteci kao i njihove trenutne dostupne količine i pregled svih registrovanih klijenata u sistemu apoteke.

Modul Klijent: pruža korisnicima mogućnost kupovine odabranih lekova i željenoj količini naravno ako je ona dostupna kao i registraciju pri istoj.

• identifikovati kako se podaci razmenjuju tj. prenose između pojedinih modula

• identifikovati koji su ulazi i koji izlazi cele aplikacije

Ulazi: odabrane opcije u vidu broja za kretanje kroz aplikaciju, pristupni pin farmaceuta, naziv leka koji se kupuje, količina traženog leka, broj zdravstvene knjižice klijenta (i ime i prezime pri registraciji).

Izlaz: liste lekova, klijenata, račun sa svim stavkama i ukupnom cenom, obrađene runtime greške

• identifikovati i napraviti test-slučajeve

Primer integracionog testiranja sa test slučajevima je u test paketu priloženog projekta, u ***klasi IntegracionoTestiranje***.

Slika prikaza putanje integracionog testiranja sistema koristeći top-down tehniku:

