KONFLIKTNE SITUACIJE

# Na jednu žalbu može da odgovori samo jedan administrator

Opis situacija:

Amidnistrator sistema pregleda sve žalbe koji su klijenti napisali. Unošenjem odgovora  
na žalbu I klikom na dugme za slanje, šalje se zahtev ka serveru za update ove žalbe.  
Server zatim pronađe žalbu na koju je administrator sistema odgovarao, promeni  
approved Boolean vrednost na false (iz razloga sto je na pocetku vrednost postavljena na true)I čuva u bazu.

Problem:

Dva administratora sistema – admin1 I admin2 kliknu na istu žalbu I skoro u isto vreme  
kliknu na dugme za slanje odgovora. Međutim, jedan od njih je prvi to uradio, pa će njegov  
odgovor biti prosleđen, dok će drugom administratoru to biti onemogućeno

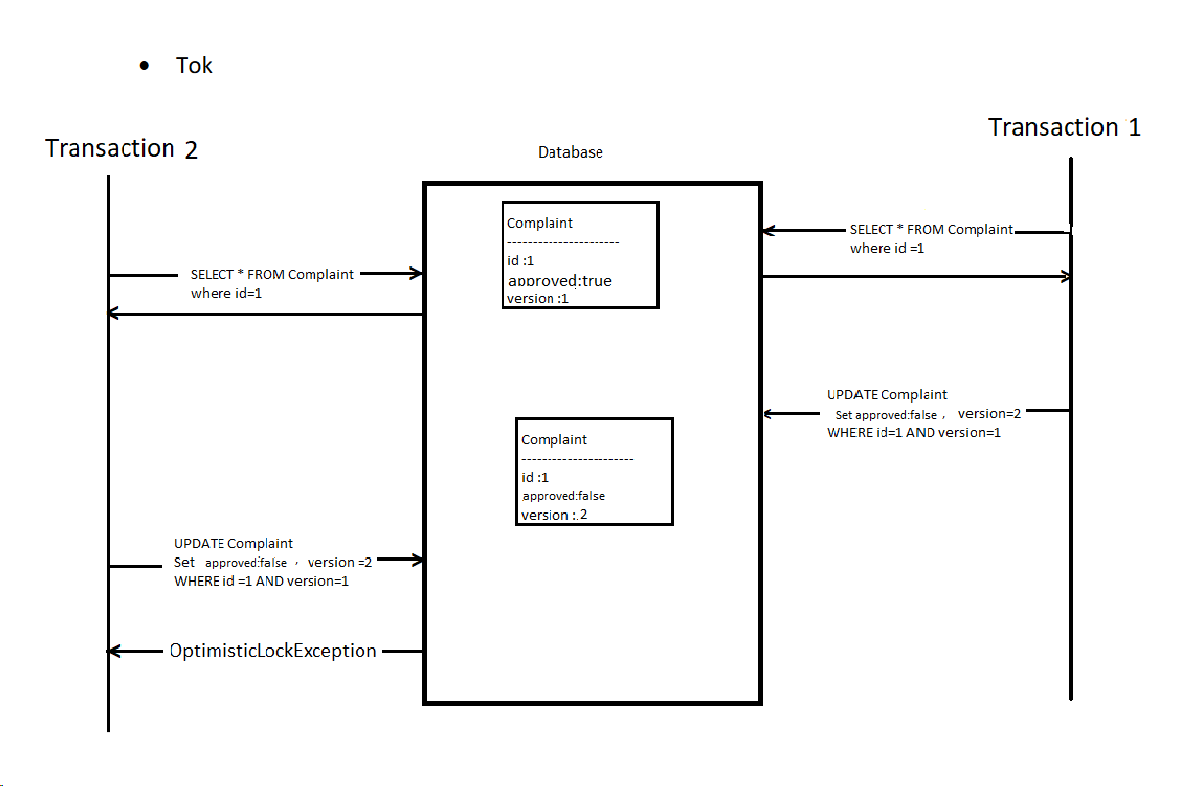
Rešenje problema:

Problem je rešen pomoću optimistic locking-a. U Complaint je dodato novo polje, private  
Long version I dodeljena mu je anotacija @Version.

Pomoću @Version anotacije postiže se to, da se pre čuvanja izmene da polje boolean  
approved dobija vrednost false, proveri verzija podatka koji se čuva. Ako je verzija podatka  
ista, kao u trenutku kada je učitana iz baze, podatak se ažurira a polje version se  
inkrementira.

U slučaju da admin1 malo brže klikne na zahtev za slanje odgovora, njegov  
zahtev će pre stići do servera, pri čemu će se njegova izmena sačuvati i njegov će odgovor  
biti poslat. A za admina2 će se proveravati verzija podatka, i ona neće biti ista kao u  
trenutku preuzimanja informacija iz baze, jer je admin1 u međuvremenu uradio izmenu.  
Iz tog razloga dolazi do ObjectOptimisticLockingFailureException-a kada se u okviru  
ClientComplaintService klase, u metodi update pozove complaintRepository.save(complaint).

Tok:



1. U jednom trenutku, samo jedan system admin može da izmeni loyalty program

Opis situacije:

Amidnistrator sistema pregleda trenutni loyalty program. Unošenjem novih vrednosti,  
šalje se zahtev ka serveru za izmenu programu. Server zatim pokuša da sačuva izmenjene  
vrednosti u bazu

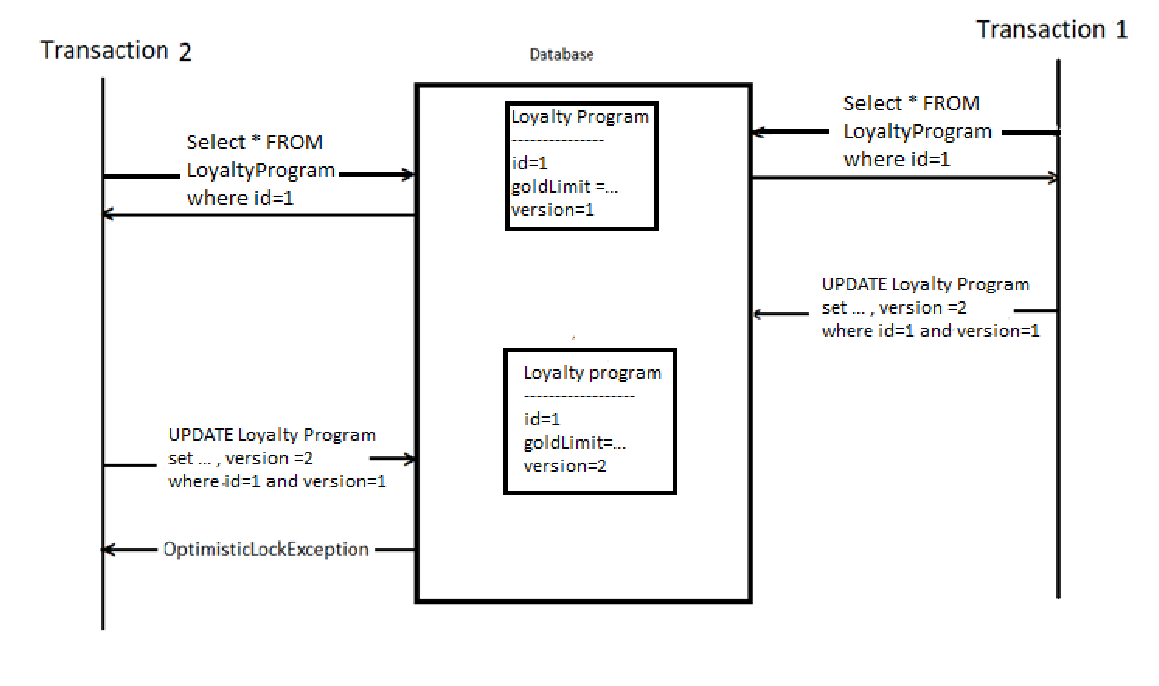
Problem:  
Dva administratora sistema – admin1 I admin2 pregledaju trenutni loyalty program I skoro  
u isto vreme kliknu na dugme za izmenu. Međutim, jedan od njih je prvi to uradio, pa će  
njegova izmena biti sačuvana, dok će drugom administratoru to biti onemogućeno.

Rešenje:

Problem je rešen pomoću optimistic locking-a. U Loyalty program je dodato novo polje,  
private Long version I dodeljena mu je anotacija @Version

Problem je rešen pomoću optimistic locking-a. U Loyalty program je dodato novo polje,  
private Long version I dodeljena mu je anotacija @Version.  
Pomoću @Version anotacije postiže se to, da se pre čuvanja izmenjnih vrednosti  
programa, proveri verzija podatka koji se čuva. Ako je verzija podatka ista, kao u trenutku  
kada je učitana iz baze, podatak se ažurira a polje version se inkrementira. U slučaju da  
admin1 malo brže klikne na zahtev za slanje odgovora, njegov zahtev će pre stići do  
servera, pri čemu će se njegova izmena sačuvati i njegov će odgovor biti poslat. A za  
admina2 će se proveravati verzija podatka, i ona neće biti ista kao u trenutku preuzimanja  
informacija iz baze, jer je admin1 u međuvremenu uradio izmenu. Iz tog razloga dolazi do  
ObjectOptimisticLockingFailureException-a kada se u okviru LoyaltyProgramService klase,  
u metodi update pozove loyaltyProgram.save (loyaltyProgram).

Tok:



# Na jedan zahtev za brisanje može da odgovori samo jedan administrator

Opis situacije:

Ukoliko neki od korisnika želi da obriše svoj nalog on šalje zahtev za brisanje. Administratori  
sistema mogu da vide sve zahteve za brisanje koji se nalaze u sistemu. Zahteve mogu da odobre ili  
da odbiju, ukoliko se zahtev odbije salje se odgovor korisniku, zašto je odbijen.

Problem:  
Dva administratora sistema – admin1 I admin2 pregledaju trenutne zahteve za brisanje I skoro  
u isto vreme kliknu na dugme za odobravanje brisanje korisnika. Međutim, jedan od njih je prvi to uradio, pa će njegova izmena biti sačuvana, dok će drugom administratoru to biti onemogućeno.

Rešenje:

Problem je rešen pomoću optimistic locking-a. U modelu Account Deletion Request je dodato novo polje, private Long version I dodeljena mu je anotacija @Version

Pomoću @Version anotacije postiže se to, da se pre čuvanja izmenjnih vrednosti  
programa, proveri verzija podatka koji se čuva. Ako je verzija podatka ista, kao u trenutku  
kada je učitana iz baze, podatak se ažurira a polje version se inkrementira. U slučaju da  
admin1 malo brže klikne na zahtev za slanje odgovora, njegov zahtev će pre stići do  
servera, pri čemu će se njegova izmena sačuvati i njegov će odgovor biti poslat. A za  
admina2 će se proveravati verzija podatka, i ona neće biti ista kao u trenutku preuzimanja  
informacija iz baze, jer je admin1 u međuvremenu uradio izmenu. Iz tog razloga dolazi do  
ObjectOptimisticLockingFailureException-a kada se u okviru Admin service klase,  
u metodi acceptAccountDeleted I unacceptAccountDeleted pozove userRepository.save(user);

Tok:

