Uputstvo za implementaciju:

HQL: http://www.tutorialspoint.com/hibernate/hibernate_query_language.htm

NOVO - Obavezno koristiti named parametre: http://www.mkyong.com/hibernate/hibernate-parameter-binding-examples/

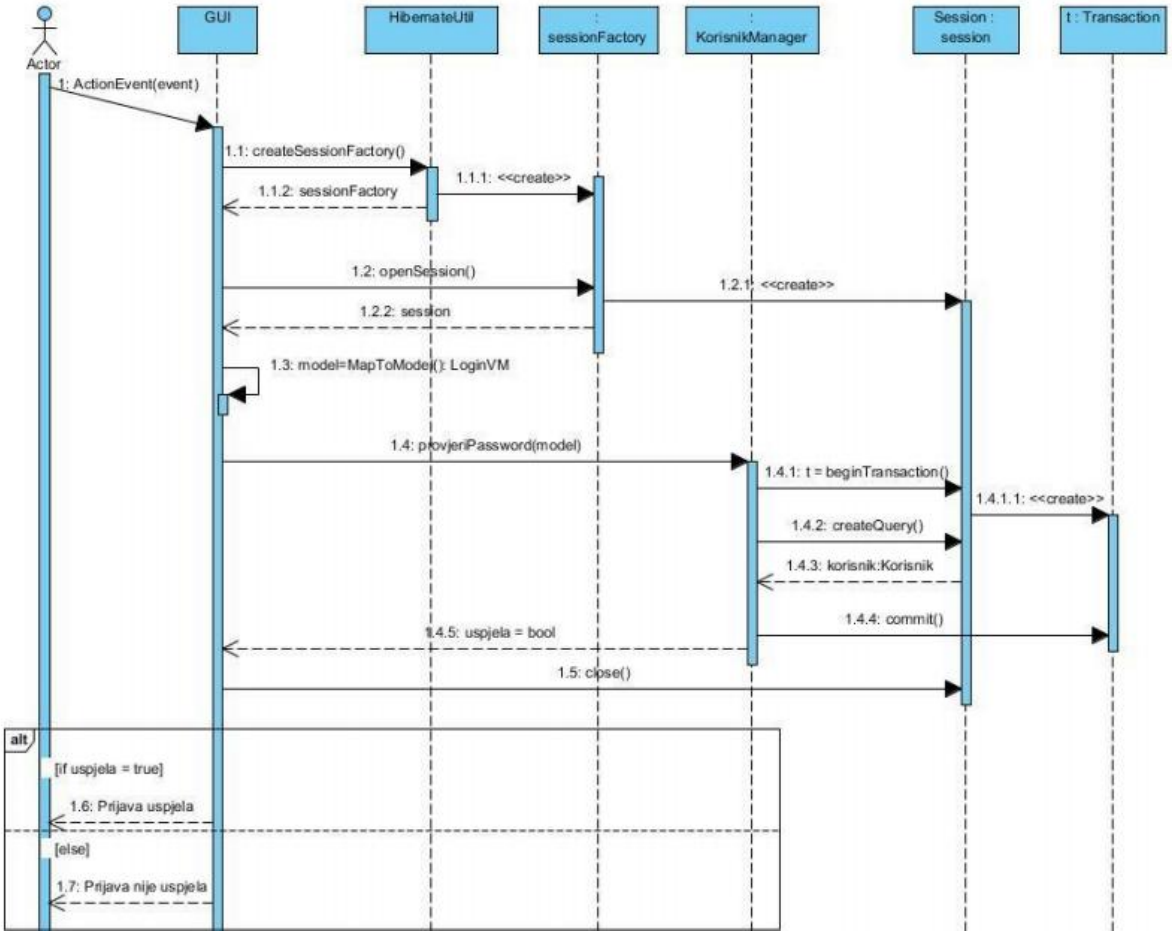
1. Instalirajte bazu... uputstvo je dato u nekom prošlom postu. Pročitajte i komentare ispod i provjerite radi li vam Hibernate i baza. Pazite da ime baze bude kao u uputstvu i ime Usera i password njegov. Pogledajte i MaterijalTest da vidite otprilike što ima.
2. Prvo da vidimo kako se radi s Hibernateom (vjerojatno već znate):

```
// BMT idiom
Session sess = factory.openSession();
Transaction tx = null;
try {
   tx = sess.beginTransaction();

   // do some work
   ...

   tx.commit();
}
catch (RuntimeException e) {
   if (tx != null) tx.rollback();
   throw e; // or display error message
}
finally {
   sess.close();
}
```

3. Kako mi radimo s tim?
    a. U GUI-u u nekom eventu napravimo Sessiju koristeći factory pomću HibernateUtil klase
    b. Instanciramo BLL klasu i session šaljem kroz konstruktor
        i. U BLL konstruktoru session spašavamo kao privatni atribut
    c. Pozovete BLL metodu nad tom klasom s parametrim koje pokupite s GUI-a, ako ima više parametara moguće je da pokupljene podatke GUI-a smestite u ViewModel i njega pošaljete
        i. U BLL metodi uradite *beginTransaction()*
        ii. Sad radite što trebate: *save, get, update, delete, createQuery*
        iii. *transaction.Commit()*;
        iv. Ako je potrebno da unutar BLL-a koristite drugu BLL klasu, to radite nakon Commita. I session joj poroslijedite kroz konstruktor i pozovete metodu kako već treba
        v. NE zatvarate sesiju
        vi. Vratite ViewModel
    d. Zatvoreti sesiju *session.Close()*

e. Prikažete podatke koje ste dobili u ViewModelu

4. Pogledajte Dijagram Sekvence: LOGIN



Implementacija Logina

Klasa za Login je KorisnikManager, a metoda ***boolean*** *provjeriPassword(LoginVM)*. Prvo napravimo konstruktor kako smo gore napisali i importujemo klasične stvari za Hibernate:

```
4
5  import org.hibernate.Transaction;
6  import org.hibernate.Session;
7
8
9  public class KorisnikManager {
10
11     private Session session;
12
13     public KorisnikManager(Session session) {
14         this.session = session;
15     }
```

Sada implementiramo metodu.

- Imporujemo Query: **import** org.hibernate.Query;
- Napravimo Query. PAZITE, ime tabele je Case Sensitive, vjerojatno i imena kolona! Ime tabele je isto kao DomainModel.
- Naše username u WHERE dio ubacujemo pomoću Named Parametara, sigurnije je, pomoću
- **import** java.util.List;
- Rezultate kverija dobijemo s q.list();
- Provjerimo da li uopće ima korisnika s tim username-om. **if**(l.size() < 0)
- Provjerimo, ako ima usera, da li je password korektan.
    - l.get(0) vraća nam prvi član ali kao objekt. Moramo Castati u string (String) l.get(0)
    - U Javi objekte poredimo s metodom *equals*. Ako koristimo == poredit će se po referenci(da li pokazuju na isti objekt), a nas zanima da li su stringo po SADRŽAJU jednaki.
    - Ukoliko su jednaki vratit će se **true**, ukoliko nisu **false**

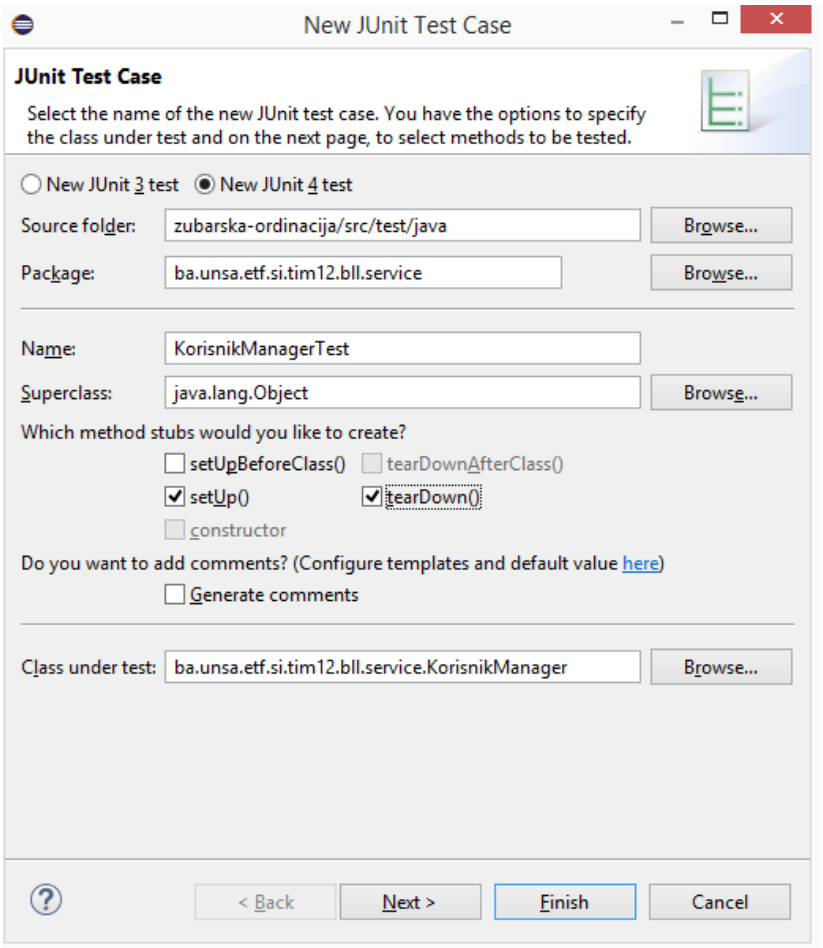
```
public boolean provjeriPassword(LoginVM model) {

    Transaction t = session.beginTransaction();

    String hql = "SELECT k.password FROM Korisnik k " +
                 "WHERE k.username = :username";
    Query q = session.createQuery(hql);
    q.setParameter(":username", model.getUsername());
    List l = q.list();

    if(l.size() < 0)
        return false;

    return model.getUsername().equals((String) l.get(0));
}
```

Nasli smo grešku u JUnit testiranju. U setParameter ide „username“. U hql stringu ostaje „:username“

Junit test klasa

Sada ćemo napraviti testnu klasu kako bi provjerili da li radi napisana metodu, i svaki naredni put to provjerimo samo jednim klikom a ne testiranjem GUI-a. **Testna metoda će biti implementirana slično kao event u GUI-u.** Naravno izuzev setUp() i tearDown() klase pomoću kojih ćemo postaviti i izbrisati testne podatke iz baze.

- Desni klik na KorisnikManager -> New-> JUnit Test Case
- Source folder promijenimo u: zubarska-ordinacija/src/**test**/java. Pošto će nam trebati označimo setUp() i tearDown() klase.



- Ako poglemo implementaciju koju smo napravili potrebno je da imamo 3 testa: pravilan username i pravilan password, pravilan username i pogrešan password, pogrešan username

```
package ba.unsa.etf.si.tim12.bll.service;

import static org.junit.Assert.*;

public class KorisnikManagerTest {

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void ProvjeriPasswordSIspravnimUsernameomIPasswordom() {
        fail("Not yet implemented");
    }

    @Test
    public void ProvjeriPasswordSIspravnimUsernameomPogresnimPasswordom()
        fail("Not yet implemented");
    }

    @Test
    public void ProvjeriPasswordSPogresnimPassworom() {
        fail("Not yet implemented");
    }
```

- Napravit ćemo testnog usera u setUp metodi i na kraju u tearDown ćemo ga obrisati

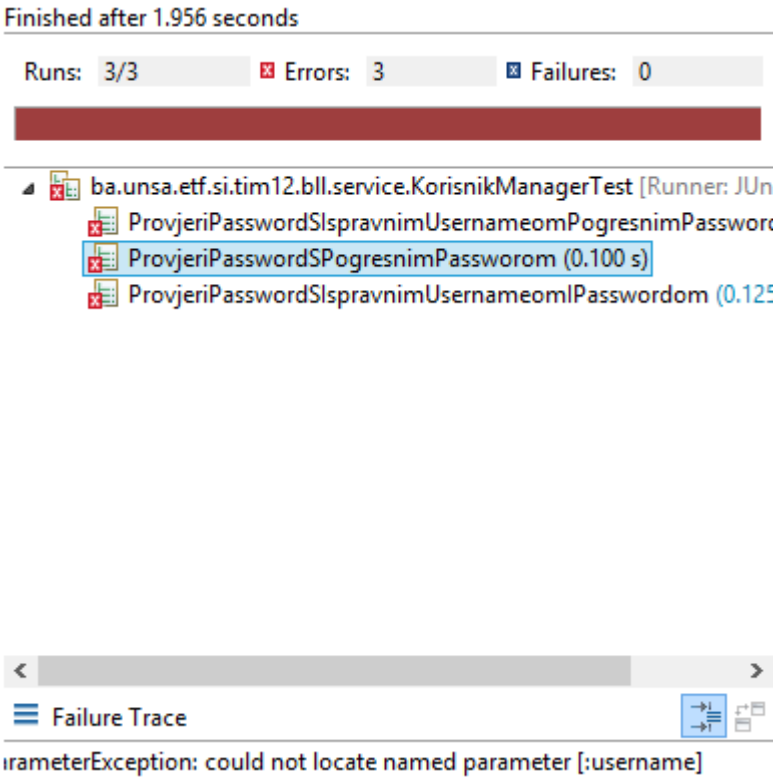
```
public class KorisnikManagerTest {

    Korisnik korisnik;
    @Before
    public void setUp() throws Exception {
        korisnik = new Korisnik();
        korisnik.setUsername("testUser1");
        korisnik.setPassword("testPassword123");

        Session sess = HibernateUtil.getSessionFactory().openSession();

        Transaction t = sess.beginTransaction();
        sess.save(korisnik);
        t.commit();

        sess.close();
    }

    @After
    public void tearDown() throws Exception {
        Session sess = HibernateUtil.getSessionFactory().openSession();

        Transaction t = sess.beginTransaction();
        sess.delete(korisnik);
        t.commit();

        sess.close();
    }
```

- Sada ćemo implementirati testne metode. Pozivanje iz GUI-a izgledat će slično.
  Evo jedna metoda ostale možete vidjeti u kodu:

```
@Test
public void ProvjeriPasswordSPogresnimPassworom() throws Exception {
    Session sess = null;

    LoginVM vm = new LoginVM();
    vm.setUsername(korisnik.getUsername()+"Greska");
    vm.setPassword(korisnik.getPassword());


    try{
        sess = HibernateUtil.getSessionFactory().openSession();

        KorisnikManager kManager = new KorisnikManager(sess);

        boolean result = kManager.provjeriPassword(vm);

        assertEquals("Korsnikov username ne postoji.", false, result);

    } catch(Exception e){
        e.printStackTrace();
        //u GUIu ne bi proslijedili nego bi nešto uradili s njim
        throw e;
    } finally {
        if(sess != null)
            sess.close();
    }

}
```

- Odete na Run as... JUnit Test i pogledate rezultate

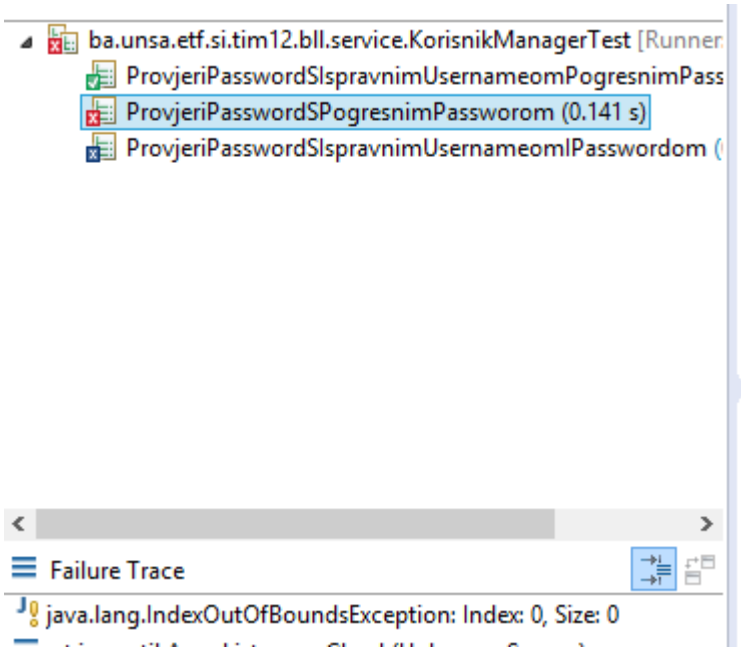
- Hop! Dobili smo grešku. Dole možete vidjeti koju.



- Promijenimo u (bez „:“)

```
25          Query q = session.createQuery(hql);
26          q.setParameter("username", model.getUsername());
27          List l = q.list();
```

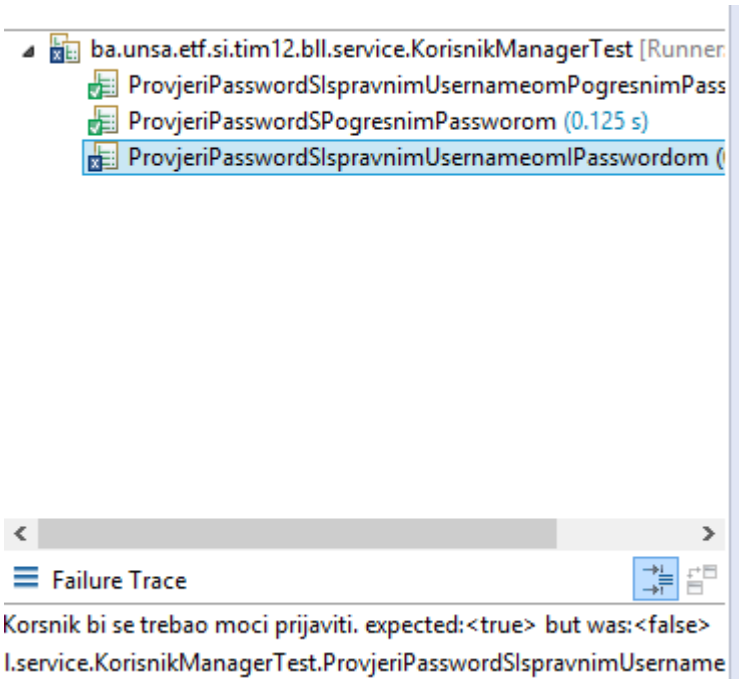
- Retestiramo: još jedna greška



- Ispravimo na l.size < **1**

```
26          q.setParameter("usern
27          List l = q.list();
28
29          if(l.size() < 1)
30              return false;
```

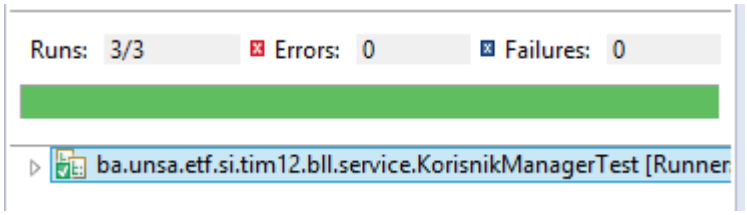
- Retestiramo:



- Gledamo jedno pola sata. Napravimo dodatnu Unit test klasu i skontamo da smo uradili glupost i umjesto .getPassword() napisali .getUsername na samom kraju. Ispravimo to i imamo ovaj rezultat

```
    public boolean provjeriPassword(LoginVM model) {

        Transaction t = session.beginTransaction();

        String hql = "SELECT k.password FROM Korisnik k " +
                     "WHERE k.username = :username";
        Query q = session.createQuery(hql);
        q.setParameter("username", model.getUsername());
        List l = q.list();

        if(l.size() < 1)
            return false;


        return model.getPassword().equals((String) l.get(0));
    }
```

- Retestiramo – Sve prolazi. Napokon.



Ko bi rekao da će ovakva glupost od implementacije imati 3 greške i koštati me bar 2 sata debagiranja.