UNIVERZITET U SARAJEVU

ELEKTROTEHNIČKI FAKULTET SARAJEVO

DOMAĆA ZADAĆA 3

VERIFIKACIJA I VALIDACIJA SOFTVERA

Studenti: Mašović Haris, Muminović amir

Indeksi: 17993, 17744 respektivno

Odsjek: Računarstvo i Informatika

Datum:

04.01.2019

Potpisi:

Zadatak 2

a) U kratkim tezama napisati scenariji testiranja i pisanja koda opisane metode.

Na osnovu odabranog kriterija (umjetnik,album ili žanr) (1) potrebno je proći kroz kolekciju nekog korisnika i utvrditi top 10 listu umjetnika, albuma ili žanrova na koje je taj korisnik potrošio najveću sumu novca. (2)

Vidimo da imamo 2 stvari koje naša metoda mora zadovoljiti, shodno tome krenimo prvo sa (1).

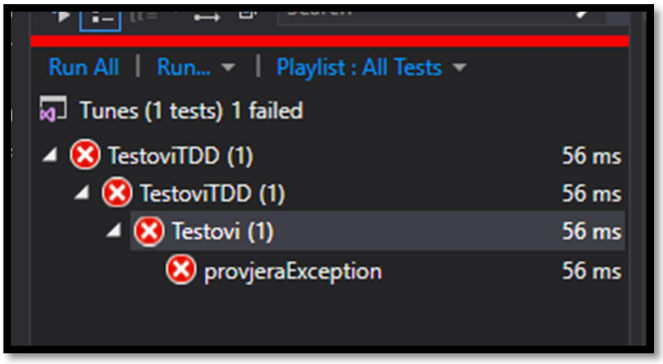
Na samom početku kreirajmo našu metodu koja će obaviti traženu funkcionalnost pod (1). Testirati ćemo našu funkcionalnost sa 2 testa. Početna metoda je sljedeća:

```
List<String> listaTopD = null;

// lista koja sadrzi top 10 po kriteriju koji se specificira
// vraca top 10 po kriteriju umjetnik, album ili zanr zavisno jel 'u', 'a' ili 'z'
        public List<String> VratiTopDeset(String kriterij)
        {
            listaTopD = new List<String>();
            return listaTopD;
        }
```

Sada kad imamo ovakvu napisanu metodu, možemo pokrenuti neki test i očekivati da on padne tj. red faza:

```
[TestMethod]
        [ExpectedException(typeof(Exception), "Nije dobro specificiran kriterij.")]
        public void provjeraException()
        {
            RegisteredMember noviKorisnik = new RegisteredMember("Haris", "Masovic",
DateTime.Now, "4012888888881881", "hmasovic1", "password");
            List<String> vracenaVrijednost1 =
noviKorisnik.VratiTopDeset("nestolijevo");
        }
```

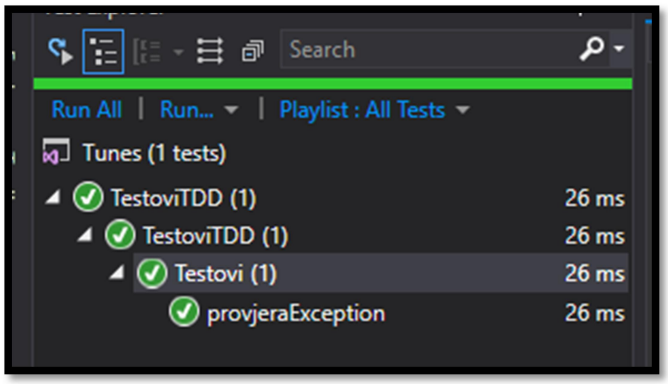
U ovom testu se očekuje exception, kao rezultat dobijamo:



Zadovoljimo našu metodu da zadovoljava tačku (1), možemo ovo uraditi na više načina, jedan od tih je sljedeći:

```
public List<String> VratiTopDeset(String kriterij)
        {
            listaTopD = new List<String>();
            throw new Exception("Nije specificiran kriterij.");
        }
```

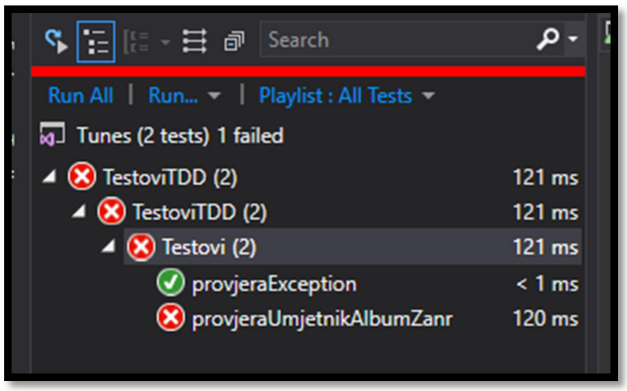
Sada smo dobili nasu green fazu:



Vidimo da je naš test prošao, optimizacije ne možemo nikakve uraditi, shodno time nastavljamo sa novom metodom:

```
[TestMethod]
        public void provjeraUmjetnikAlbumZanr()
        {
            RegisteredMember noviKorisnik = new RegisteredMember("Haris", "Masovic",
DateTime.Now, "4012888888881881", "hmasovic1", "password");
            try
            {
                List<String> vracenaVrijednost1 = noviKorisnik.VratiTopDeset("u");
                List<String> vracenaVrijednost2 = noviKorisnik.VratiTopDeset("a");
                List<String> vracenaVrijednost3 = noviKorisnik.VratiTopDeset("z");
                Assert.AreEqual(true, true);
            }
            catch(Exception)
            {
                Assert.AreEqual(true, false);
            }
        }
```

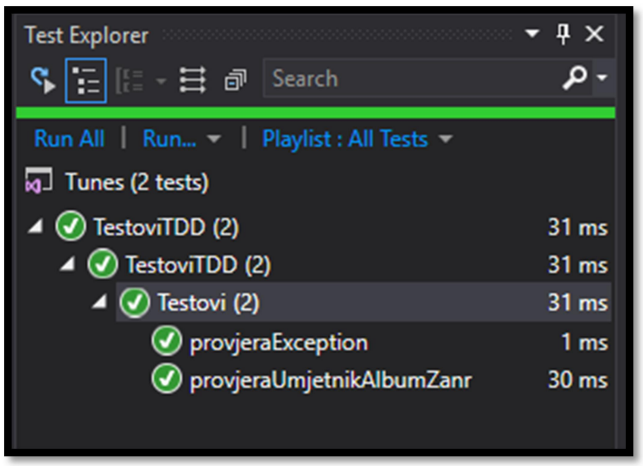
Kao rezultat dobijamo sljedeće:



Vidimo da je naša red faza zadovoljena, napišimo sada našu metodu da zadovolji ovaj test:

```
public List<String> VratiTopDeset(String kriterij)
        {
            List<String> listaTopD = new List<String>();
            if (kriterij == "u" || kriterij == "a" || kriterij == "z")
            {
                return listaTopD;
            }
            else throw new Exception("Nije specificiran kriterij.");
        }
```

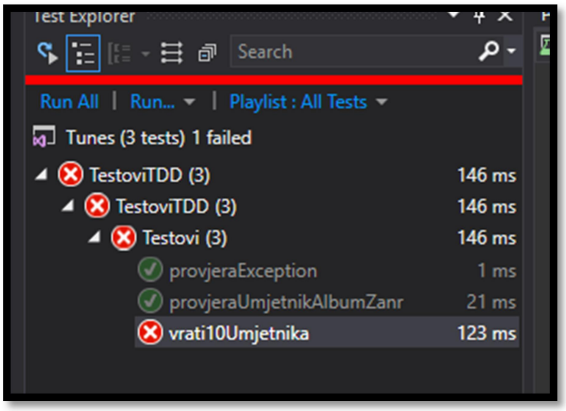
Kao rezultat prepravke imamo našu green fazu:



Vidimo da ne možemo nikakve refaktoringe uraditi, shodno tome prelazimo na novi test gdje ćemo testirani novu implementaciju koda za našu tačku (2) u kojoj ćemo testirati funkcionalnost sa 3 testa:

```
[TestMethod]
        public void vrati10Umjetnika()
        {
            RegisteredMember noviKorisnik = new RegisteredMember("Haris", "Masovic",
DateTime.Now, "4012888888881881", "hmasovic1", "password");
            // koristimo pomocni stub pomocu kojeg generisemo podatke za testiranje
            PomocniStub novi = new PomocniStub();
            List<String> rez = novi.odradiUmjetnike(noviKorisnik);
            List<String> vracenaVrijednost1 = noviKorisnik.VratiTopDeset("u");

            CollectionAssert.AreEqual(rez, vracenaVrijednost1);
        }
```

PomocniStub će biti poslije objašnjenjen pod c.



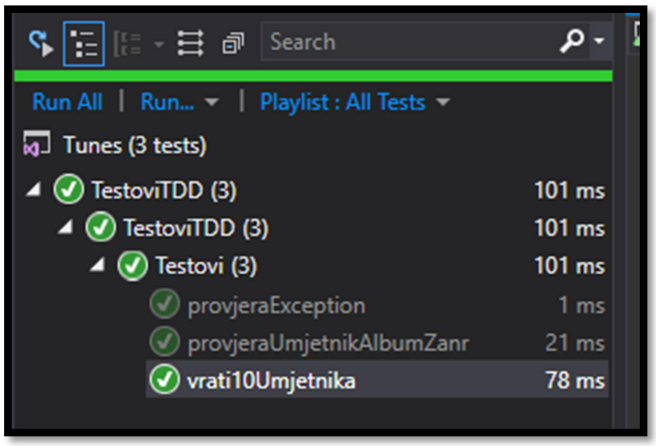
Vidimo da naš novonapisani test ne prolazi, shodno tome, uradimo implementacijski dio:

Da bi rješili ovaj test, morat ćemo kreirati novu klasu koju ćemo koristiti u našoj implementaciji i jednu privatnu metodu, čitav novi kod je u nastavku prikazan:

```
#region TDD_NOVA_METODA

        List<String> listaTopD = null;
// lista koja sadrzi top 10 po kriteriju koji se specificira
        List<Autor> autori;

        private void odradiUmjetnike()
        {
            autori = new List<Autor>();

            foreach(Tune element in mojaBiblioteka)
            {
                Autor a = new Autor();
                a.naziv = element.Artist;
                a.cijena = element.Price;
                if (autori.Find(x => x.naziv == a.naziv) == null) {
                    autori.Add(a);
                }
                else autori.Find(x => x.naziv == a.naziv).cijena += a.cijena;
            }
        }



        // vraca top 10 po kriteriju
        public List<String> VratiTopDeset(String kriterij)
        {
            listaTopD = new List<String>();
            if (kriterij == "u" ) // trazimo top 10 umjetnika
            {
                odradiUmjetnike();
                autori = autori.OrderByDescending(x => x.cijena).ToList();
                var prvihDeset = autori.Take(10);
                foreach(var element in prvihDeset)
                {
                    listaTopD.Add(element.naziv);
                }
            }
            else if(kriterij == "a")
            {
```

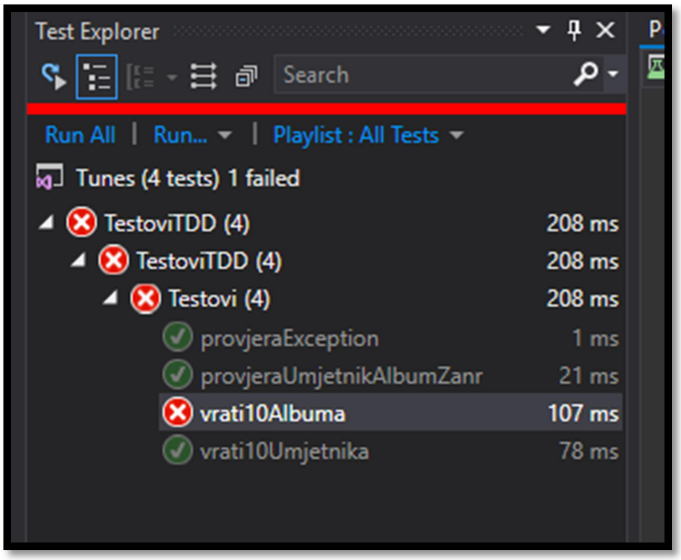
```
            }
            else if(kriterij == "z")
            {

            }
            else throw new Exception("Nije specificiran kriterij.");
            return listaTopD;
        }

        #endregion

private class Autor
        {
            public String naziv;
            public double cijena;
        }
```

Sada kada smo napisali našu novu metodu, testiranjem ćemo dobiti našu green fazu. Vidimo da zasad ne možemo nikakva poboljšanja napraviti.



Prelazimo na naš novi test:

```
[TestMethod]
        public void vrati10Albuma()
        {
            RegisteredMember noviKorisnik = new RegisteredMember("Haris", "Masovic",
DateTime.Now, "4012888888881881", "hmasovic1", "password");
            // koristimo pomocni stub pomocu kojeg generisemo podatke za testiranje
            PomocniStub novi = new PomocniStub();
            List<String> rez = novi.odradiAlbume(noviKorisnik);
            List<String> vracenaVrijednost1 = noviKorisnik.VratiTopDeset("a");

            CollectionAssert.AreEqual(rez, vracenaVrijednost1);
        }
```

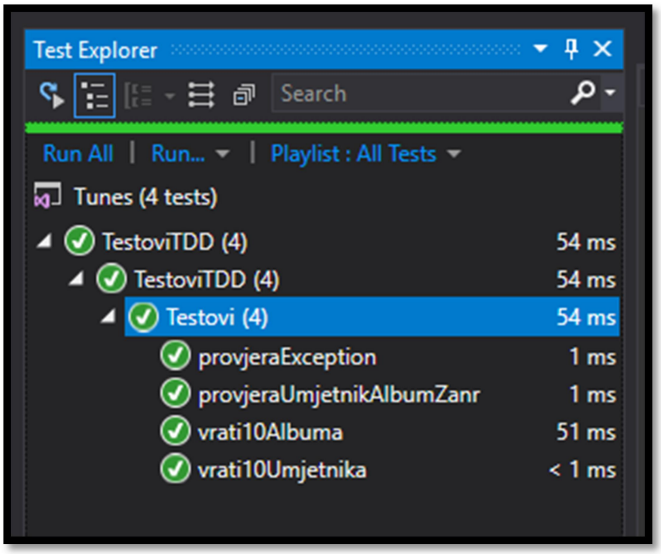


Novo nastali kod i dodatna implementacija:

```
List<Album> albumi;

        private void odradiAlbume()
        {
            albumi = new List<Album>();

            foreach (Tune element in mojaBiblioteka)
            {
                Album a = new Album();
                a.naziv = element.Album;
                a.cijena = element.Price;
                if (albumi.Find(x => x.naziv == a.naziv) == null)
                {
                    albumi.Add(a);
                }
                else albumi.Find(x => x.naziv == a.naziv).cijena += a.cijena;
            }
        }


        // vraca top 10 po kriteriju
        public List<String> VratiTopDeset(String kriterij)
        {
            listaTopD = new List<String>();
            if (kriterij == "u" ) // trazimo top 10 umjetnika
            {
                odradiUmjetnike();
                autori = autori.OrderByDescending(x => x.cijena).ToList();
                var prvihDeset = autori.Take(10);
                foreach(var element in prvihDeset)
                {
                    listaTopD.Add(element.naziv);
                }
            }
            else if(kriterij == "a")
            {
                odradiAlbume();
                albumi = albumi.OrderByDescending(x => x.cijena).ToList();
                var prvihDeset = albumi.Take(10);
```

```
            foreach (var element in prvihDeset)
            {
                listaTopD.Add(element.naziv);
            }
        }
        else if(kriterij == "z")
        {

        }
        else throw new Exception("Nije specificiran kriterij.");
        return listaTopD;
    }

    #endregion

    private class Album
    {
        public string naziv;
        public double cijena;
    }
```

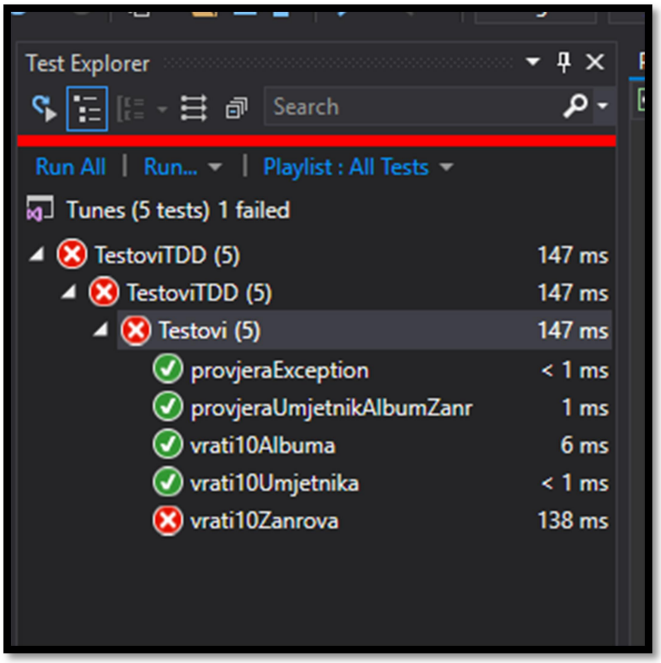
Sada kada opet pokrenemo naše testove imamo sljedeći rezultat, u ovom trenutku možemo napraviti optimizaciju, ali ćemo uraditi i slučaj žanrova pa ćemo spojiti to zajedno:



Na kraju još ostaje da napišemo treći test za slučaj žanra:

```
[TestMethod]
    public void vrati10Zanrova()
    {
        RegisteredMember noviKorisnik = new RegisteredMember("Haris", "Masovic",
DateTime.Now, "4012888888881881", "hmasovic1", "password");
        // koristimo pomocni stub pomocu kojeg generisemo podatke za testiranje
        PomocniStub novi = new PomocniStub();
        List<String> rez = novi.odradiZanrove(noviKorisnik);
        List<String> vracenaVrijednost1 = noviKorisnik.VratiTopDeset("z");

        CollectionAssert.AreEqual(rez, vracenaVrijednost1);
    }
```

Pokrenemo li naš novonapisani ujedno i zadnji test imamo sljedeće:

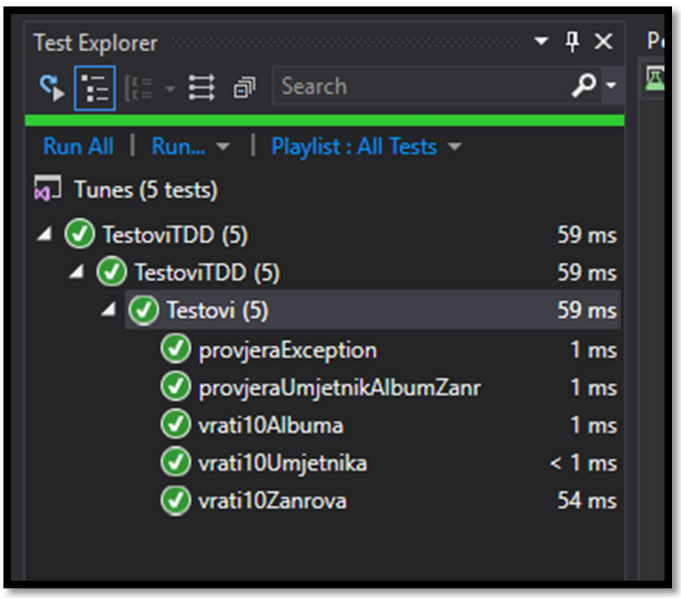


Prelazimo na našu green fazu, novonastali dodatni i kod naše metode je sljedeći:

```
List<Zanr> zanrovi;

        private void odradiZanrove()
        {
            zanrovi = new List<Zanr>();

            foreach (Tune element in mojaBiblioteka)
            {
                Zanr a = new Zanr();
                a.naziv = element.Genre;
                a.cijena = element.Price;
                if (zanrovi.Find(x => x.naziv == a.naziv) == null)
                {
                    zanrovi.Add(a);
                }
                else zanrovi.Find(x => x.naziv == a.naziv).cijena += a.cijena;
            }
        }


        // vraca top 10 po kriteriju
        public List<String> VratiTopDeset(String kriterij)
        {
            listaTopD = new List<String>();
            if (kriterij == "u" ) // trazimo top 10 umjetnika
            {
                odradiUmjetnike();
                autori = autori.OrderByDescending(x => x.cijena).ToList();
                var prvihDeset = autori.Take(10);
                foreach(var element in prvihDeset)
                {
                    listaTopD.Add(element.naziv);
                }
```

```
            }
            else if(kriterij == "a")
            {
                odradiAlbume();
                albumi = albumi.OrderByDescending(x => x.cijena).ToList();
                var prvihDeset = albumi.Take(10);
                foreach (var element in prvihDeset)
                {
                    listaTopD.Add(element.naziv);
                }
            }
            else if(kriterij == "z")
            {

                odradiZanrove();
                zanrovi = zanrovi.OrderByDescending(x => x.cijena).ToList();
                var prvihDeset = zanrovi.Take(10);
                foreach (var element in prvihDeset)
                {
                    listaTopD.Add(element.naziv);
                }
            }
            else throw new Exception("Nije specificiran kriterij.");
            return listaTopD;
        }

        private class Zanr
        {
            public String naziv;
            public double cijena;
        }
```

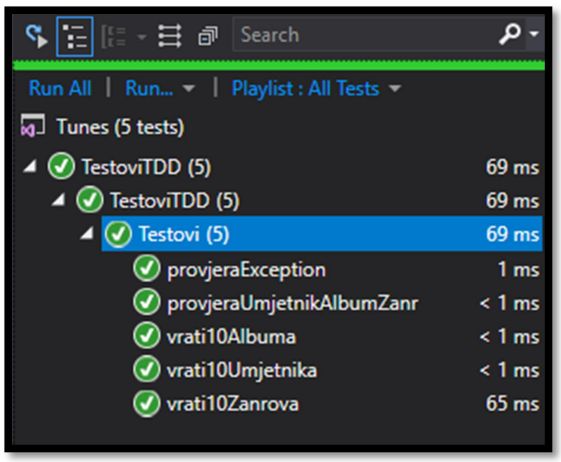
Ponovnim pokretanjem naše metode imamo sljedeće:



Vidimo da naša metoda radi ono što je traženo u postavci. Na kraju možemo uraditi refaktoring, gdje ćemo naše navedene privatne metode ujediniti u jednu, kao finalni kodni rezultat sveukupni za našu novu metodu imati ćemo manje linija koda i bolju efikasnost:

```
#region TDD_NOVA_METODA

        List<String> listaTopD = null;
       // lista koja sadrzi top 10 po kriteriju koji se specificira
        List<Objekat> objekti;

        private void odradi(String y)
        {
            objekti = new List<Objekat>();
            foreach(Tune element in mojaBiblioteka)
            {
                Objekat a = new Objekat();
                if (y == "u") a.naziv = element.Artist;
                else if (y == "a") a.naziv = element.Album;
                else if (y == "z") a.naziv = element.Genre;
                a.cijena = element.Price;
                if (objekti.Find(x => x.naziv == a.naziv) == null) objekti.Add(a);
                else objekti.Find(x => x.naziv == a.naziv).cijena += a.cijena;
            }
        }

        // vraca top 10 po kriteriju
        public List<String> VratiTopDeset(String kriterij)
        {
            listaTopD = new List<String>();
            if (kriterij == "u" || kriterij == "a" || kriterij == "z")
              // trazimo top 10 umjetnika
            {
                odradi(kriterij);
                objekti = objekti.OrderByDescending(x => x.cijena).ToList();
                var prvihDeset = objekti.Take(10);
                foreach (var element in prvihDeset)
                {
                    listaTopD.Add(element.naziv);
                }
            }
            else throw new Exception("Nije specificiran kriterij.");
            return listaTopD;
        }

        private class Objekat
        {
            public String naziv;
            public double cijena;
        }
#endregion
```

Ponovnim pokretanjem testova, vidimo da testovi i dalje prolaze:



b) Tri testne metode koje će u potpunosti testirati navedenu metodu su već urađene pod a), a to su vrati10Albuma, vrati10Umjetnika, vrati10Zanrova, može se u tu skupinu dodati i provjeraException pa zajedno sa tim ćemo imati 100% code coverage.

c) Na kraju ostaje još da objasnimo naš dodatni stub koji generiše naše vrijednosti za testiranje. Stubovi se inače koriste za generisanje podataka tj. dummy objekata koji nama fale. Pošto mi imamo već sve urađeno kad je u pitanju naš registrovani korisnik (implementacija) mi smo iskoristili ovaj stub da generišemo podatke koji će nam trebati prilikom testiranja. Svaka metoda ovog stuba će generisati određene podatke klase Registrovanog korisnika i vratiti rezultat koji je očekivan.

```
class PomocniStub
    {
// metoda za generisanje umjetnika i vraca rezultat koji se mora dobiti
public List<String> odradiUmjetnike(RegisteredMember noviKorisnik)
        {
            for (int i = 0; i < 15; ++i)
            {
                Tune nova = new Tune("prva", "saban" + i.ToString(), "sabanov prvi",
"narodna", 10.0, ".mp3", 2400, 256, 10 + i);
                noviKorisnik.MojaBiblioteka.Add(nova);
            }

            for (int i = 0; i < 5; ++i)
            {
                Tune nova = new Tune("prva" + i.ToString(), "jana", "sabanov prvi",
"narodna", 10.0, ".mp3", 2400, 256, 20);
                noviKorisnik.MojaBiblioteka.Add(nova);
            }
List<String> vraceno = new List<String>();
            vraceno.Add("jana");
            vraceno.Add("saban14");
            vraceno.Add("saban13");
            vraceno.Add("saban12");
            vraceno.Add("saban11");
            vraceno.Add("saban10");
            vraceno.Add("saban9");
            vraceno.Add("saban8");
            vraceno.Add("saban7");
            vraceno.Add("saban6");
```

```
            return vraceno;
        }
// metoda koja se koristi za generisanje albuma i vraca rezultat koji se mora dobiti
        public List<String> odradiAlbume(RegisteredMember noviKorisnik)
        {
            for (int i = 0; i < 15; ++i)
            {
                Tune nova = new Tune("prva", "saban", "sabanov prvi" + i.ToString(),
"narodna", 10.0, ".mp3", 2400, 256, 10 + i);
                noviKorisnik.MojaBiblioteka.Add(nova);
            }

            for (int i = 0; i < 5; ++i)
            {
                Tune nova = new Tune("prva" + i.ToString(), "jana", "janin",
"narodna", 10.0, ".mp3", 2400, 256, 20);
                noviKorisnik.MojaBiblioteka.Add(nova);
            }

            List<String> vraceno = new List<String>();
            vraceno.Add("janin");
            vraceno.Add("sabanov prvi14");
            vraceno.Add("sabanov prvi13");
            vraceno.Add("sabanov prvi12");
            vraceno.Add("sabanov prvi11");
            vraceno.Add("sabanov prvi10");
            vraceno.Add("sabanov prvi9");
            vraceno.Add("sabanov prvi8");
            vraceno.Add("sabanov prvi7");
            vraceno.Add("sabanov prvi6");
            return vraceno;
        }

// metoda koja se koristi za generisanje zanrova i vraca rezultat koji se mora dobiti
        public List<String> odradiZanrove(RegisteredMember noviKorisnik)
        {
            for (int i = 0; i < 15; ++i)
            {
                Tune nova = new Tune("prva", "saban", "sabanov prvi", "narodna" +
i.ToString(), 10.0, ".mp3", 2400, 256, 10 + i);
                noviKorisnik.MojaBiblioteka.Add(nova);
            }

            for (int i = 0; i < 5; ++i)
            {
                Tune nova = new Tune("prva" + i.ToString(), "jana", "janin",
"veselija", 10.0, ".mp3", 2400, 256, 20);
                noviKorisnik.MojaBiblioteka.Add(nova);
            }

            List<String> vraceno = new List<String>();
            vraceno.Add("veselija");
            vraceno.Add("narodna14");
            vraceno.Add("narodna13");
            vraceno.Add("narodna12");
            vraceno.Add("narodna11");
            vraceno.Add("narodna10");
            vraceno.Add("narodna9");
            vraceno.Add("narodna8");
            vraceno.Add("narodna7");
            vraceno.Add("narodna6");
            return vraceno; } }
```