

Datum: 03.05.2018.

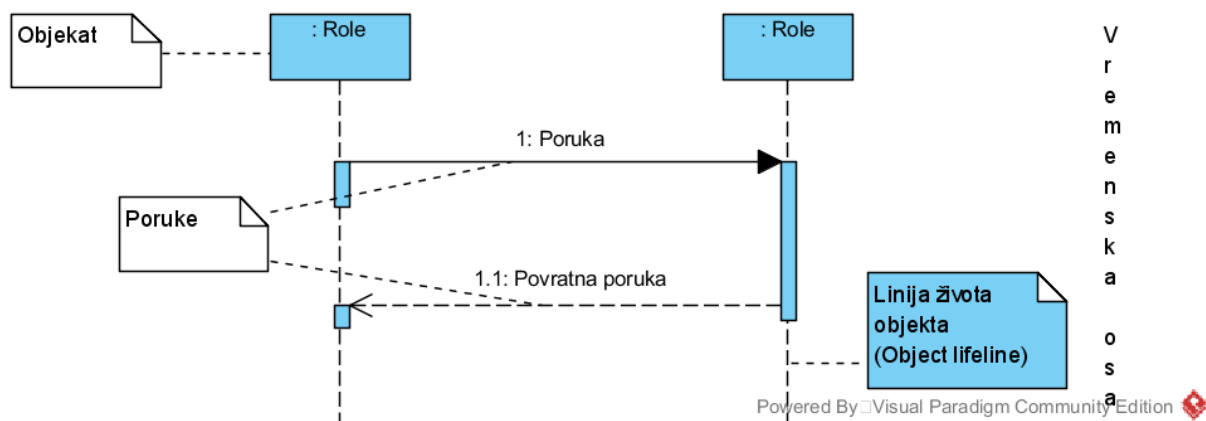
UML dijagrami interakcije

UML dijagrami interakcije su: **sekvencijalni dijagram** (sequence diagram) i **kolaboracijski dijagram**. UML koristi dijagrame interakcije za opis *interakcije između objekata*. Objekti međusobno komuniciraju putem *poruka*. Interakcija se koristi za opis dinamičkih aspekata sistema.

Sekvencijalni dijagrami

Sekvencijalni dijagram (sequence diagram) je dio UML notacije i koristi se za grafički prikaz operacija sistema hronološkim redoslijedom.

Sekvencijalni dijagram je dio objektno orijentiranog modela. Predstavlja dinamički pogled na objekte (instance klase) i poruke koje objekti međusobno razmjenjuju. Sekvencijalni dijagram može sadržavati učesnike (actors) koji su u interakciji sa sistemom. Interakcija se predstavlja u dvije dimenzije; vertikalnom dimenzijom predstavljamo vrijeme, a u horizontalnoj predstavljamo objekte.



Sekvencijalni i kolaboracijski dijagrami predstavljaju iste informacije na dva različita načina (semantički su ekvivalentni). Osnovna razlika je uključenje vremenske dimenzije u kontekst sekvencijalnog dijagrama.

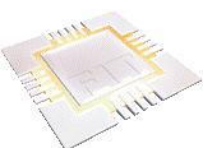
Zašto razvijamo sekvencijalne dijagrame?

Sekvencijalni dijagram prikazuje ponašanje objekata u terminima interakcije. Dodatak je dijagramu klasa koji prikazuje statičku strukturu sistema. Sekvencijalni dijagram određuje ponašanje klasa, interfejsa i načine korištenja njihovih operacija.

Razvoj sekvencijalnog dijagrama, također nam pruža mogućnost testiranja statičkog modela na konceptualnom nivou, jer istinski može predstaviti scenarij u kojem su klase iz dijagrama klasa instancirane u objekte potrebne za izvršenje scenarija.

Sekvencijalnim dijagramima pojednostavljujemo hronološki opis uloga objekta.

Razvoj svakog modela započinje razvojem use case dijagrama. Use case dijagrame koristimo za identifikaciju koncepata za domain model. Svakom use case-u odgovara jedan ili više sekvencijalnih dijagrama. Obično imamo jedan sekvencijalni dijagram za



glavni tok događaja i po jedan sekvencijalni dijagram za svaki alternativni tok događaja slučaja upotrebe.

Aktivnosti izrade sekvencijalnog dijagrama su:

1. definiranje aktera i objekata,
2. definiranje poruka,
3. definiranje indikatora sinhronizacije.

1. Definiranje objekata

Objekt je instanca jedne klase. Objekti se u sekvencijalnim dijagramima predstavljaju pravougaonikom („**box**“) i vertikalnom isprekidanom linijom („**lifeline**“).



Osnovne osobine objekata su prikazane tabelom 1:

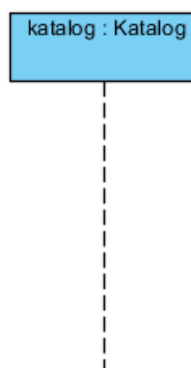
Osobina	Opis
Name	Naziv objekta (neobavezno)
Description	Opisni komentari o objektu
UML Stereotype	Proširenje semantike objekta izveden iz postojećih objekata.
Class name	Klasa ili interfejs čija je instanca objekat.

Tabela 1: Osobine objekata

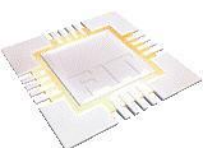
Možemo imati neimenovane instance klase ili interfejsa, ali ih moramo pridružiti nekoj od klase ili interfejsa u *Class name* box-u.

Objekti se imenuju na sljedeći način:

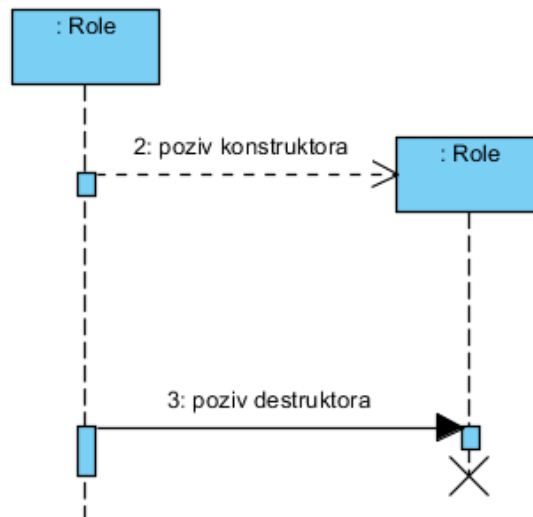
naziv_objekta:Naziv_klase



Objekat *katalog* je instanca klase *Katalog*



Većina objekata postoji dok traje interakcija, ali objekti mogu nastati i u toku interakcije. Njihov životni vijek počinje nakon prijema poruke <<create>>. Prijem poruke <<create>> nije dozvoljen učesnicima (akterima). Objekti mogu biti uništeni u toku interakcije. Njihov životni vijek se završava nakon prijema poruke <<destroy>> ili <<self-destroy>>. Poruka *self-destroy* prikazuje autodestrukciju objekta (objekt uništava sam sebe). Poruke <<destroy>> i <<self-destroy>> su posljednje poruke koje dva objekta mogu razmijeniti u sekvencijalnom dijagramu i ne mogu biti rekurzivne.



Konstruktor postavlja instancu objekta na odgovarajuće mjesto na vremenskoj liniji, a destruktork završava životnu liniju objekta.

Aktiviranje objekata se predstavlja *uskim pravougaonikom* na liniji objekta, a predstavlja trajanje operacije (akcije) koju objekat obavlja. Ukoliko imate potrebu, pored aktivacije možete dodati zabilješku (engl. note).

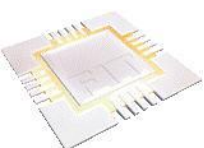
Sekvencijalni dijagram se formira tako što se prvo na vrh dijagrama duž ose X, postavite objekti koji učestvuju u interakciji. Obično se objekti koji započinju interakciju pozicioniraju krajnje lijevo, a svaki sljedeći objekt desno od prethodnog (s lijeva na desno). Nakon toga poruke koje ovi objekti šalju i primaju smještaju se duž ose u rastućem nizu po vremenskom redoslijedu od vrha ka dnu. Ovo pruža jasnu asocijaciju na tok kontrole u protoku vremena.

2. Definiranje učesnika

Sistem koji opisujemo je u interakciji sa učesnicima (actors). Učesnici se definiraju na sličan način kao i u dijagramima slučajeva korištenja. U sekvencijalnim dijagramima učesnici imaju svoju životnu liniju.

Učesnik je najčešće inicijator interakcije i prvi na horizontalnoj osi.

Isti učesnici mogu učestvovati u više use-case ili sekvencijalnih dijagrama ukoliko imaju istu ulogu. Učesnici su zajednički za sve dijagrame modela koji ih koriste.



Osobine učesnika se ne razlikuju značajno od osobina objekata. Za učesnike se ne definiše klasa (*Class name*), jer oni nisu objekti sistema.

Učesnici u sekvencijalnom dijagramu, kao i objekti, mogu se kreirati na dva načina:

1. koristeći alatni okvir i
2. prenošenjem sa liste učesnika iz Explorer-a.

3. Definiranje poruka

Objekti međusobno komuniciraju putem poruka koje prenose informacije s ciljem izvršenja određene aktivnosti. U UML notaciji, poruke se predstavljaju strelicama, koje polaze sa linije života jednog objekta ili učesnika, a završavaju na liniji života drugog objekta ili učesnika. Rekurzivna poruka je prikazana strelicom koja polazi i završava na liniji života istog objekta ili učesnika.

Poruka (Message) je definirana nazivom i argumentima. Možemo imati jedan ili više argumenata. Poruka je opisana na sljedeći način:

Return value = Naziv_poruke (arguments, ...)

Uz naziv poruke se navodi i broj kojim se definira redoslijed izvršenja poruka. Lista parametara nije obavezna, kao ni zagrade, ukoliko je lista parametara prazna.

Argumenti se navode u zagradama, a uvjeti prijelaza u uglastim zagradama. Poruke imaju pošiljaoca, primaoca i operaciju. Pošiljalac i primalac poruke je učesnik ili objekt. Prijem poruke predstavlja stimulans za izvršenje operacije koja ima određeni ishod.

Poruke također možemo definirati između dva aktera, ali ćemo dobiti upozorenje za ovu poruku pri provjeri modela.

Akcija	Opis
Poziv (call)	<i>Pokreće operaciju objekta primaoca (može pozivati i sam sebe)</i>
Povratak (return)	<i>Vraća vrijednost primaocu (može biti rekurzivna)</i>
Konstruktor (create) <<create>> →	<i>Kreira se objekt</i>
Destruktor (destory) <<destory>> →	<i>Uništava se objekat (objekat može uništiti sam sebe)</i>

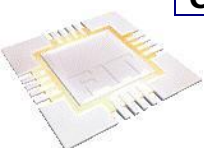
Tabela 2: Vrste poruka

Istu poruku je moguće koristiti u više sekvencijalnih dijagrama. Kopirana poruka ima isti naziv koji možete promijeniti. Svaka promjena operacije ili kontrolnog toka ima uticaj na sve dijagrame u kojima se ova poruka koristi.

Sekvencijalni broj poruke je isti u svim dijagramima koji koriste tu poruku.

Osobine poruke (dostupne unutar Power Designer-a):

Osobina	Opis
Name	Naziv poruke
Comment	Opisni komentari o poruci



Stereotype	Proširenje semantike poruke izvedeno iz postojećih poruka.
Sender	Objekt koji šalje poruku
Receiver	Objekt koji prima poruku
Sequence number	Omogućuje nam poruci dodijeliti sekvencijalni broj.
Reverse direction	Promjena smjera poruke (zamjena uloga objekata pošiljaoca i primaoca)

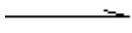
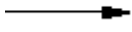
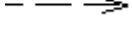

„Detail page“ poruka u sekvencijalnom dijagramu (dostupno unutar Power Designer-a):

Osobina	Opis
Action	Omogućuje prikazati poruke tipa: konstruktor, destruktor i auto-destruktor objekta.
Control flow	Prikazuje na koji način je poruka poslana.
Operation	Operacija (metoda) klase koja odgovara poruci.
Arguments	Argumenti operacije.
Return value	Povratna vrijednost operacije.
Predecessor list	Popis sekvencijalnih brojeva operacija koje se izvršavaju prije. Npr.: '1,2,4/ 3' (operacije 1,2 i 4 se izvršavaju prije operacije 3).
Condition	Uslov pridružen poruci (boolean izraz). Npr.: [dialing time < 30 sec]
Begin time End time	Koristi se za definiranje ograničenja: Npr.: Begin time = t1, End time = t2, ograničenje = {t2 - t1 < 30 sec}
Support delay	Definira kašnjenje u prijenosu (neaktivno za rekurzivne poruke). Npr.: Ukoliko Support delay nije označen, znači da je vrijeme početka i završetka izvršenja operacije isto.

Indikatori sinhronizacije poruka (Control flow)

Indikator sinhronizacije određuje tip poruke.

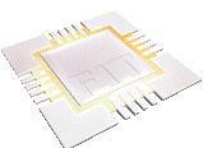
Poruke mogu imati sljedeće indikatore sinkronizacije:

Tip	Opis	Oznaka
Asynchronous	Asinhronne poruke su one koje ne zahtijevaju odgovor (nisu blokirajuće - block).	
Procedure Call	Poziv procedure je sinhrona poruka i zahtijeva odgovor. Odgovor se šalje sa odredišta po prijemu poruke i izvršenju operacije. Blokira izvršenje dok odgovor ne stigne.	
Return	Obično je pridružena pozivu proceduri i predstavlja odgovor po izvršenju operacije.	
Undefined	Nedefinirani tip poruke.	

Indikatore sinhronizacije i tipove operacija možemo kombinirati na sljedeći način:

Control flow	No action	Create	Destroy	Self-Destroy
Asynchronous	+	+	+	-
Procedure Call	+	+	+	-
Return	+	-	-	+
Undefined	+	+	+	-

+ = dozvoljeno
- = nedozvoljeno

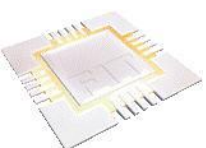
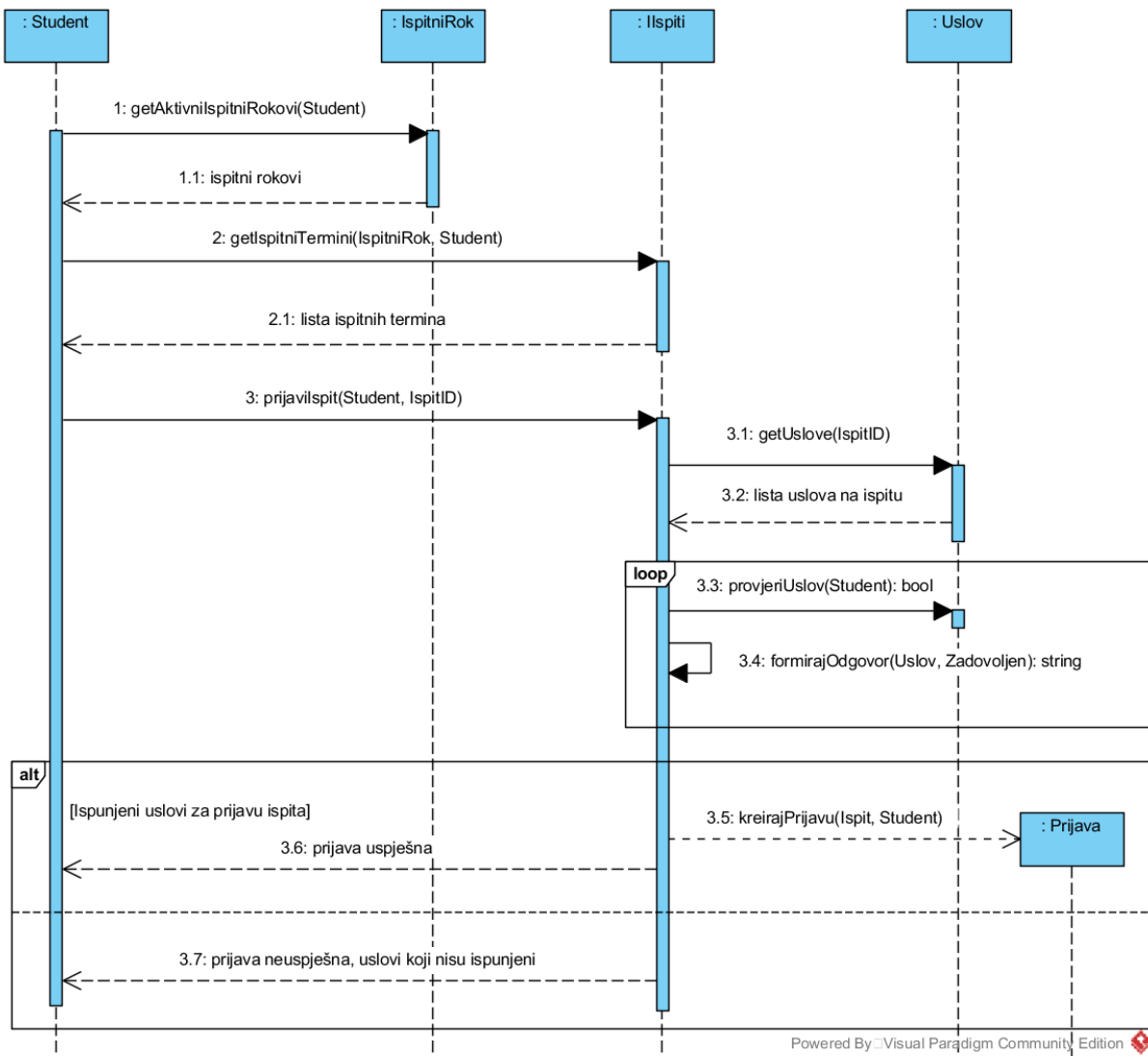


Operacije (Operation)

„Operation dropdown listbox“ nam omogućava dodijeliti poruku jednoj klasi za metodu. Poruku pridružujemo klasi objekta primaoca kao operaciju. Poruka može biti pridružena i već postojećoj operaciji. Ova opcija može biti jako korisna u procesu implementacije.

Zadaci

U nastavku je postupak prijave ispita detaljnije razrađen sekvencijalnim dijagramom. Za prijavu ispita potrebni su objekti klasa: *IspitniRok*, *Ispit*, *Uslov* i *Prijava*, gdje se objekat klase *Prijava* kreira samo ako su svi uslovi za prijavu ispita ispunjeni.



Zadatak 1: Dodjela predmeta (eSluzba)

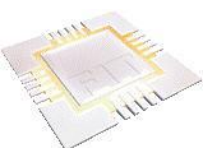
Jedna od mogućnosti referenta studentske službe je i dodjela predmeta nastavnog plana određenog studija. Za svaki predmet se prilikom dodjele definišu sljedeći podaci: tip predmeta, redni broj u semestru, semestar, šifra, ECTS, predavanja, vježbe i status. Pri tome je moguće definisati i uslovljenost, odnosno preciznije, listu predmeta koje studenti moraju položiti da bi mogli pratiti predmet koji se trenutno evidentira. Uslovljenost može biti potpuna ili djelimična. U sklopu jednog nastavnog plana nije moguće dodati isti predmet više puta. Svakom dodjelom novog predmeta referentu se osvježava lista svih predmeta nastavnog plana. Dodjela predmeta traje sve dok ukupan broj ECTS bodova ne dostigne vrijednost 180 ili 120, što zavisi od ciklusa na kojem se sluša odabrani nastavni plan (1. ciklus = 180 ECTS, 2. ciklus = 120 ECTS).

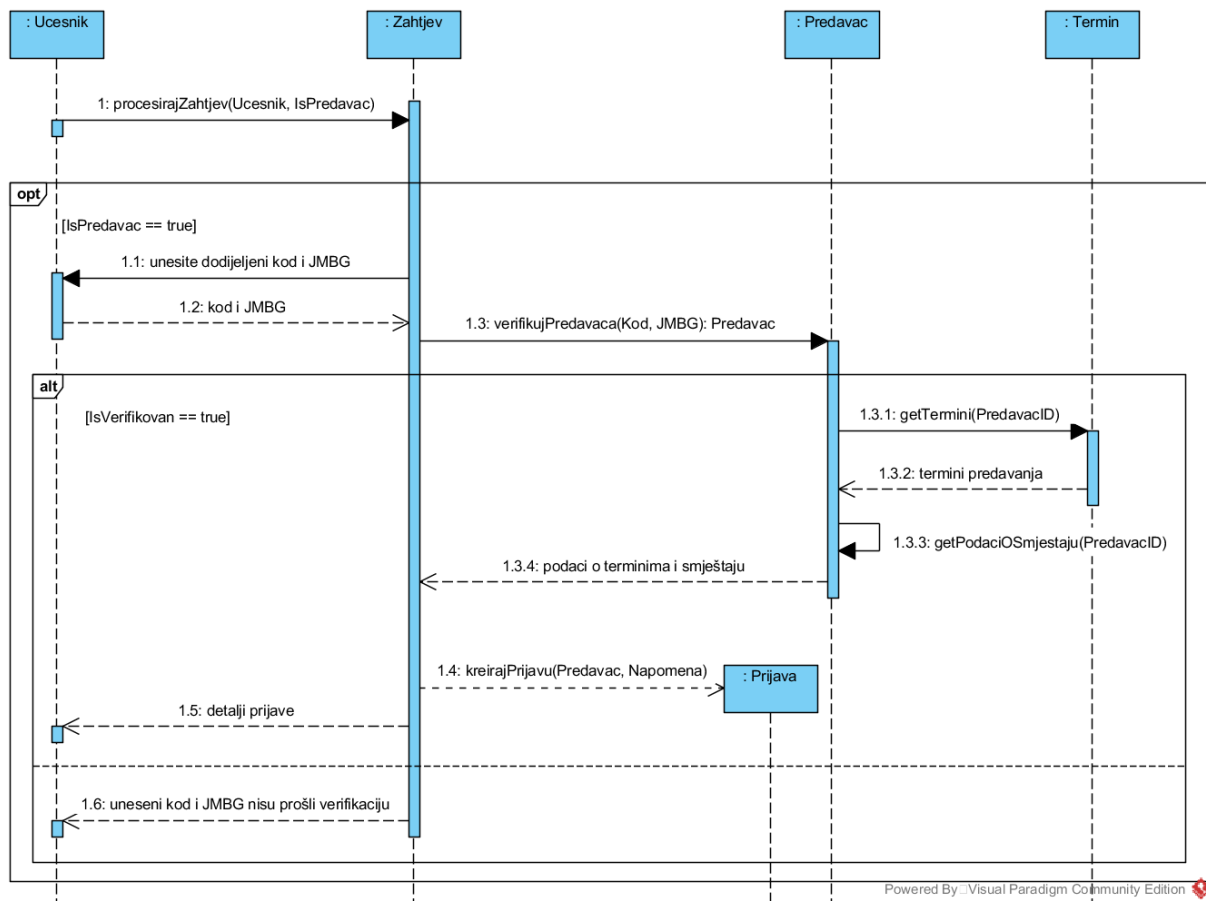
Zadatak 2: Prijava učešća na konferenciju

Koristeći sekvencijalni dijagram modelirati scenarij prijave učesnika na određenu konferenciju. Pretpostavka je da je aplikant već registrovan i prijavljen na sistem. Ukoliko se učesnik pri slanju zahtjeva prijavi kao predavač, potrebno je provjeriti da li se on nalazi na definisanoj listi predavača konferencije. U tom procesu sistem od učesnika zahtjeva unos matičnog broja i dodijeljenog koda kako bi se potvrdio njegov identitet. Uspješnom potvrdom predavaču se proslijedi raspored termina dogovorenih predavanja, te podaci o smještaju za vrijeme konferencije. Ako je riječ o učesniku koji se prijavljuje kao slušaoc na konferenciji potrebno je da se putem sistema izvrši uplata kotizacije (određenog iznosa). U tom procesu učesnik može odabrati i opciju „Uključen smještaj“, pri čemu sistem provjerava prvenstveno da li ima slobodnih mjesta. Ukoliko je smještaj dostupan, njegova cijena se zaračuna na kotizaciju i učesniku se proslijede podaci. Sve prijave učesnika je neophodno evidentirati u sistemu.

Koristiti objekte sljedećih klasa: **Ucesnik, Predavac, Zahtjev, Termin i Prijava.**

U nastavku je prikazan prijedlog rješenja koji se odnosi na prijavu učešća predavača. Za vježbu dopunite dijagram prijavom za slušaoce konferencije i dodatnim objektom klase **Smjestaj**.





Zadatak 3: Inverzni inženjering

Primjenom inverznog inženjeringa za sljedeći kod napraviti odgovarajući sekvencijalni dijagram.

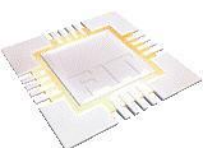
```

public class UsersController
{
    private List<Users> users = new List<Users>();

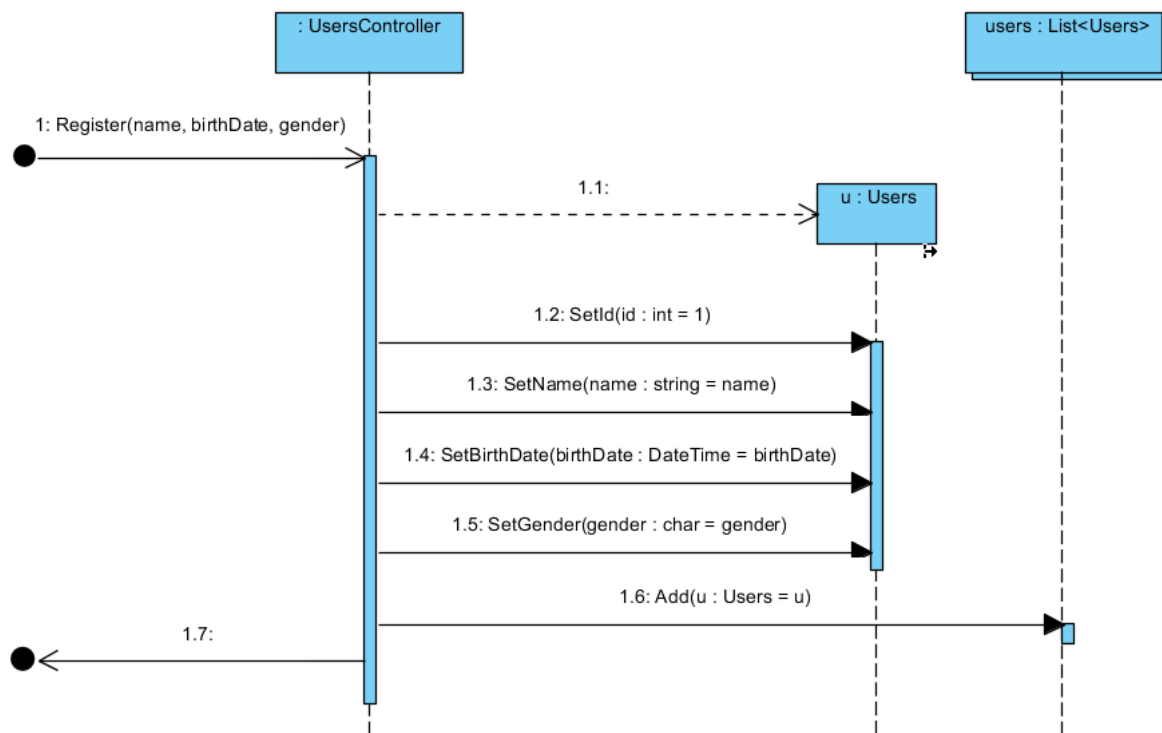
    public void Register(string name, DateTime birthDate, char gender)
    {
        Users u = new Users();

        u.SetId(1);
        u.SetName(name);
        u.SetBirthDate(birthDate);
        u.SetGender(gender);

        users.Add(u);
    }
}
    
```



Prijedlog rješenja:



Zadatak 4: Inverzni inženjering

Primjenom inverznog inženjeringa za sljedeći kod napraviti odgovarajući sekvencijalni dijagram.

```

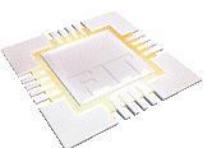
public class OrdersController
{
    private List<OrderItems> orderItems = new List<OrderItems>();

    public bool AddProduct(Product p, int quantity)
    {
        if (p.GetStockQuantity() < quantity)
            return false;

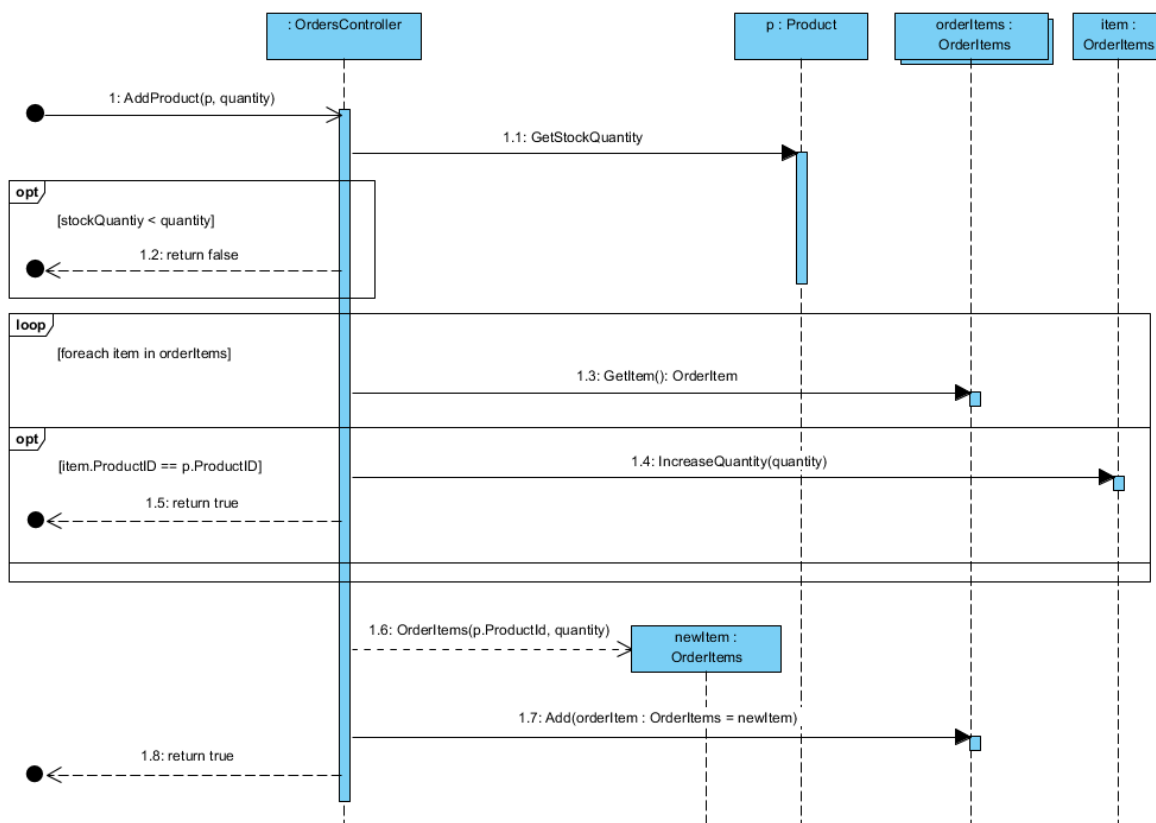
        foreach (OrderItems item in orderItems)
        {
            if (item.ProductID == p.ProductID)
            {
                item.IncreaseQuantity(quantity);
                return true;
            }
        }

        OrderItems newItem = new OrderItems(p.ProductID, quantity);
        orderItems.Add(newItem);

        return true;
    }
}
  
```



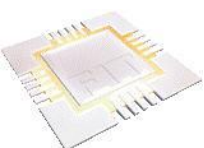
Prijedlog rješenja:



Ispitni zadatak:

U cilju realizacije što većeg broja inovativnih projekata sa fokusom na upotrebu informacijskih tehnologija u svakodnevnom životu, potrebno je implementirati softversku aplikaciju koja treba da proces prijave, odobrenja i praćenje svih faza u realizaciji projekata učini što jednostavnijim. Funkcionalni zahtjevi koje je neophodno ispuniti su dati u nastavku.

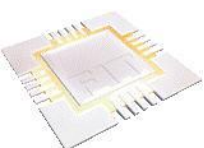
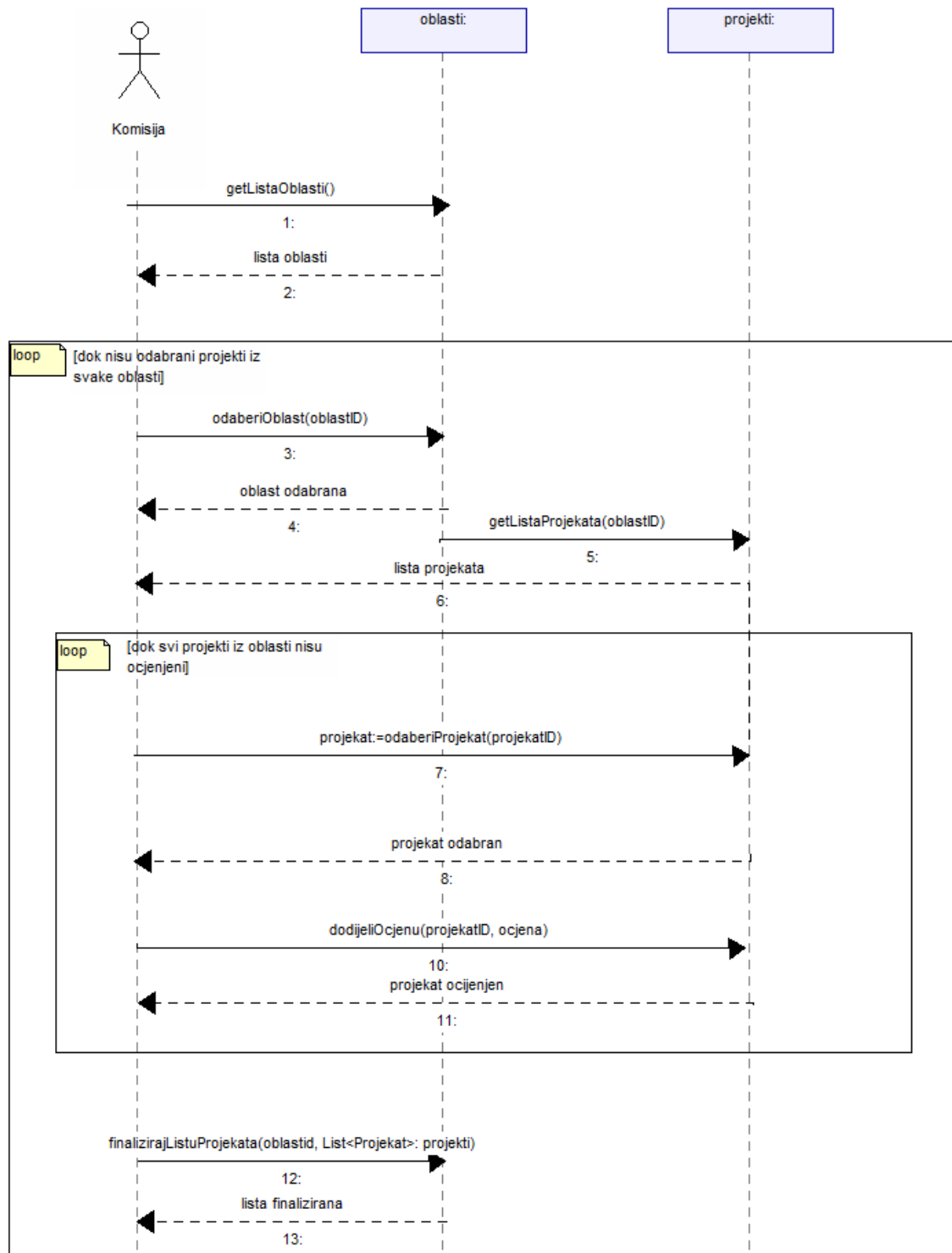
- Organizatorima omogućiti definisanje rokova (Opis, Pocetak, Kraj) za prijave projekata (Naslov, Opis, Prilog, Status) po unaprijed zadatim oblastima (Naziv, Status). U svakoj od oblasti može biti najviše 20 prijavljenih projekata. Pri svakom definisanom roku se automatski ažurira glavna sekcija novosti na stranici.
- Učesnicima (KorisnickoIme, Lozinka, Drzava) zainteresovanim za razvoj inovativnih projekata dozvoliti registraciju i prijavu na stranicu. Nakon uspješne prijave, učesnici mogu prijavljivati projekte ukoliko nije istekao definisani rok, te ukoliko nije dostignut maksimalan broj prijava u odabranoj oblasti. U sklopu prijave projekata definišu se i podaci o timu (Ime, Prezime) koji će biti uključen u samu realizaciju. Obavezno je istaknuti projektog vođu, te ostaviti njegove kontakt podatke. U projektom timu može biti angažirano maksimalno 7 učesnika.
- Organizatori (Firma, Predstavnik, Email, Telefon, WebAdresa) mogu da pretražuju arhivu prijavljenih projekata po oblastima, te ih proslijede komisiji na reviziju. Komisija (Broj, Predsjednik komisije i Dva člana) definiše po 5 projekata iz svake oblasti koji će dobiti potrebna finansijska sredstva za realizaciju. Organizator nakon toga definiše iznos za svaki od projekata u skladu sa njihovim opsegom i obavještava projektog vođu o uslovima i načinu dalje saradnje.



Sekvencijalnim dijagramom modelirati proces komisijske revizije projekata koje su izdvojili organizatori (koristiti najmanje dva objekta – Projekti i Oblasti).

Rješenje studenta:

Ispod je prijedlog rješenja studenta koji je osvojio maksimalan broj bodova (15).
Dijagram je kreiran u alatu Open ModelSphere.



Ukratko

Koraci pri izradi sekvencijalnog dijagrama: Polazimo od tekstualnoga opisa slučajeve upotrebe i konceptualnog dijagrama, a zatim koncepte u sekvencijalnom dijagramu pretvaramo u objekte. Iz tekstualnog opisa slučajeve upotrebe definiraju se događaji i operacije za objekte. Dijagram sekvenci može poslužiti da se operacije pridruže klasama na dijagramu klasa.

Prateću video lekciju možete pogledati u sklopu YouTube kanala na sljedećem linku:

- [Sekvencijalni dijagram](#) (Open ModelSphere 3.2)
- [Sekvencijalni dijagram ispitni zadatak](#) (Open ModelSphere 3.2)

Literatura

1. Martin Fowler, UML Distilled Third Edition: A brief guide to the standard object modeling language, 2004. [Download](#) (Dostupno: 03.05.2018.).
2. Craig Larman, Applyng UML and paterns, Prentice Hall, 2004.
3. <https://www.visual-paradigm.com/>
4. <https://www.youtube.com/user/VisualParadigm/>
5. <http://www.uml.org/>

