Datum: 30.03.2017.

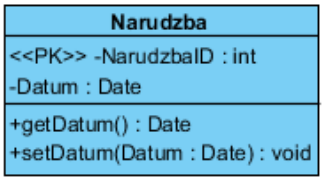
1. Dijagram klasa (Class diagram)

Dijagram klasa (class diagram) također je dio objektno orjentisanog modela i jedan od UML dijagrama. Dijagram klasa definira *statičku strukturu modela*. Čine ga skup paketa (packages), klasa (classes), interfejsa (interfaces) i veza (relationships) među njima.
Klasa opisuje skup objekata, asocijacije opisuju skup veza, dok su objekti instance klasa i veze su instance asocijacija.

1.1. DEFINISANJE KLASA

Klasa opisuje skup objekata koji imaju sličnu strukturu i ponašanje, zajedničke atribute, operacije, relacije i semantiku. Struktura je opisana atributima i asocijacijama, dok je ponašanje opisano operacijama.

Klase su osnovni gradivni blok OOM-a. Klase i relacije između njih čine osnovnu strukturu OOM-a. Klase definiraju koncepte koji mogu biti fizički (npr. automobil), poslovni (npr. narudžba), logički (npr. red vožnje) ili aplikacijski (npr. OK button).



Slika 1: Primjer klase sa atributima i operacijama

Osobine klase su prikazane tabelom 1:

Tabela 1: Osobine klase

Osobina	**Opis**
Name	Naziv klase
Description	Opisni komentari o klasi
UML Stereotype	Dodatna klasifikacija postojeće klase. Proširenje semantike objekta bez potrebe za promjenom strukture.
Classifier type	Koristi se za implementaciju.
Visibility	Opseg vidljivosti klase. Ova vrijednost određuje način na koji drugi objekti vide i koriste klasu.
Abstract	Klasa ne može biti instancirana.
Final	Klasa ne može imati nasljednike.

Klasa također posjeduje sljedeće karakteristike:

Osobina	**Opis**
Fields	*Polja*, odnosno *Atributi* definiraju karakteristike klasa.
Methods	Operacije određuju ponašanje instanci klasa.
Used by	Drugi objekti koji koriste instance klase.
Nested/Inner Classes	Definira listu sadržanih klasa u jednoj klasi.

1.2. OPSEG KLASE (class visibility)

Opseg klase je način na koji je vide drugi objekti. To može uticati na strukturu i/ili ponašanje objekta ili objekti mogu uticati na osobine klase.

Osobina	Opseg (visibility)
Public	Klasi mogu pristupiti svi objekti modela
Private	Klasa može pristupiti samo sama sebi
Protected	Klasa može pristupiti sama sebi i mogu joj pristupati njeni izvedeni objekti
Default access	Klasi mogu pristupiti svi objekti iz paketa

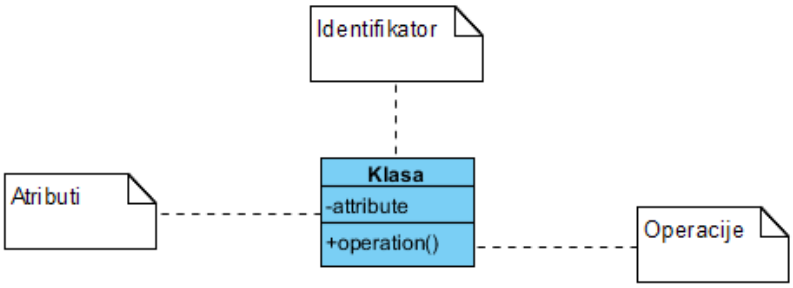
1.3. KARDINALITET ILI BROJNOST KLASA

Kardinalitet (brojnost) klasa određuje koliko objekata (instanci) može imati jedna klasa.

Kardinalitet	Broj instanci (objekata)
0..1	Od nula do jedan
0..*	Od nula do n (n>0)
1..1	Jedan i samo jedan
1..*	Od jedan do n (n>0)
*	n (n>0)

1.4. KREIRANJE KLASE

Simbol klase sadrži tri dijela: identifikator, atributi i operacije.



1.4.1. DEFINISANJE ATRIBUTA KLASE

Atributi su osobine klase ili interfejsa. Klase ne moraju imati atribute, a mogu ih imati više.

Svi objekti jedne klase imaju iste atribute, ali njihove vrijednosti mogu biti različite. Nazivi atributa jedne klase su jedinstveni, što znači da klasa ne može imati dva atributa s istim nazivom.

Možemo definirati atribute za:
- Klase
- Interfejse i
- Identifikatore.

Atributi klase i interfejsa imaju sljedeće osobine:

Osobina	**Opis**
Name	Naziv atributa
UML Stereotype	Proširenje semantike atributa izvedeno iz drugih atributa.
Visibility	Opseg (vidljivost atributa)
Static	Atribut klase. Za sve instance ima istu vrijednost.
Volatile	Nije atribut klase. Definisan je get i set operacijama.
Initial value	Početna vrijednost atributa
Domain	Naziv pridružene domene.
UML Constraints	Ograničenja u kojima se navodi i da li je atribut dio jedinstvenog identifikatora. U fizičkom modelu podataka (PMD) jedinstveni identifikator je primarni ključ tabele. Jedinstveni identifikator postoji samo u klasi (ne postoji u interfejsu).

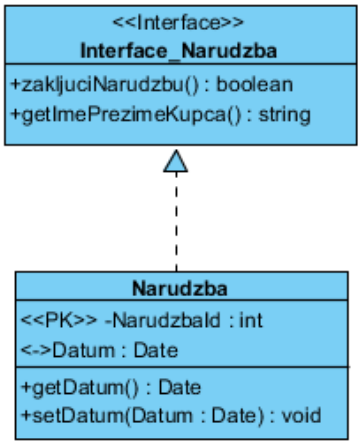
Atributu je moguće pridružiti domenu. Domena nam pruža tip podatka, karakteristike tipa podatka, kao i provjeru parametara.

Tip podatka atributa, tip podatka povratne vrijednosti operacije ili tip podatka parametra može biti tipa jedne klase.

1.5. DEFINISANJE INTERFEJSA (INTERFACES) U DIJAGRAMU KLASA

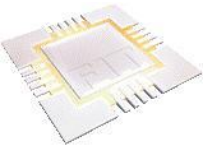
Interfejs je skup operacija koje određuju i specificiraju javno ponašanje klase. Jedna klasa može imati više interfejsa i svaki interfejs predstavlja samo jedan dio ponašanja klase. Interfejs uključuje samo deklaraciju operacija.

Atributi interfejsa se razlikuju od atributa klasa, jer atributi interfejsa mogu biti samo konstantne vrijednosti (static ili frozen).



Opseg interfejsa predstavlja pogled koji drugi objekti imaju na taj interfejs. Interfejs može uticati na strukturu i ponašanje objekata koji su u njegovom dosegu i također objekti iz opsega mogu uticati na strukturu i ponašanje interfejsa.

Jedan interfejs može koristiti atribute drugih interfejsa i klasa. Atributi se dodaju interfejsu na isti način kao i klasi.



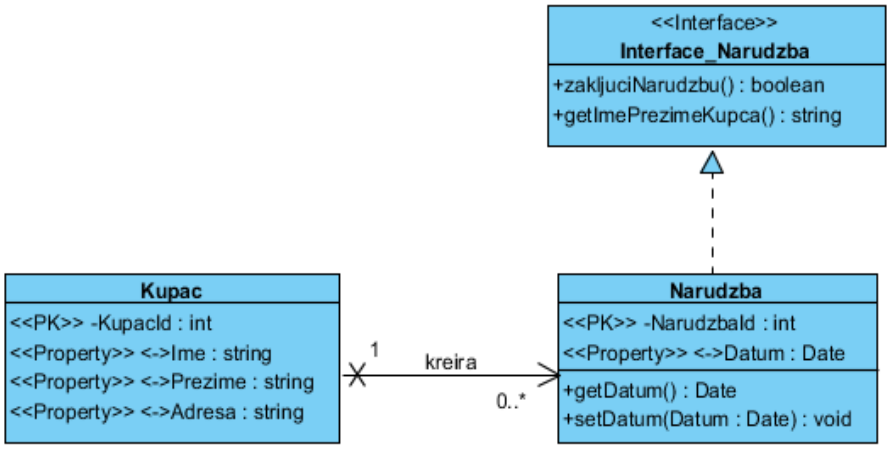
Operacija interfejsa je usluga koje mogu zatražiti objekti. Operacije se dodaju interfejsima na isti način kao i klasama.

Klasa i interface čije metode ona implementira su povezani ***vezom realizacije*.**

1.6. MIGRACIJA ATRIBUTA JEDNE KLASE

Asocijacije uzrokuju migraciju atributa klase u toku generisanja programskog koda.

Primjer:



Generisani C# kod za klasu kupac:

```
public partial class Kupac
    {
        //Fields
        private int _KupacId;
        private string _Ime;
        private string _Prezime;
        private string _Adresa;
        private Narudzba[] _Narudzba;

        //Properties
        private int KupacID
        {
            get
            {
                return _KupacID;
            }

            set
            {
                _KupacID = value;
            }
        }

        private string Ime
        {
            get
            {
                return _Ime;
              }
```

```
        set
        {
            _Ime = value;
        }
    }

    private string Prezime
    {
        get
        {
            return _Prezime;
        }

        set
        {
            _Prezime = value;
        }
    }

    private string Adresa
    {
        get
        {
            return _Adresa;
        }

        set
        {
            _Adresa = value;
        }
    }

    private Narudzba[] Narudzba
    {
        get
        {
            return _Narudzba;
        }

        set
        {
            _Narudzba = value;
        }
    }

    //Methods

} //end class
```

Asocijacija između kupca i narudžbe je predstavljena kao atribut klase *Kupac*:

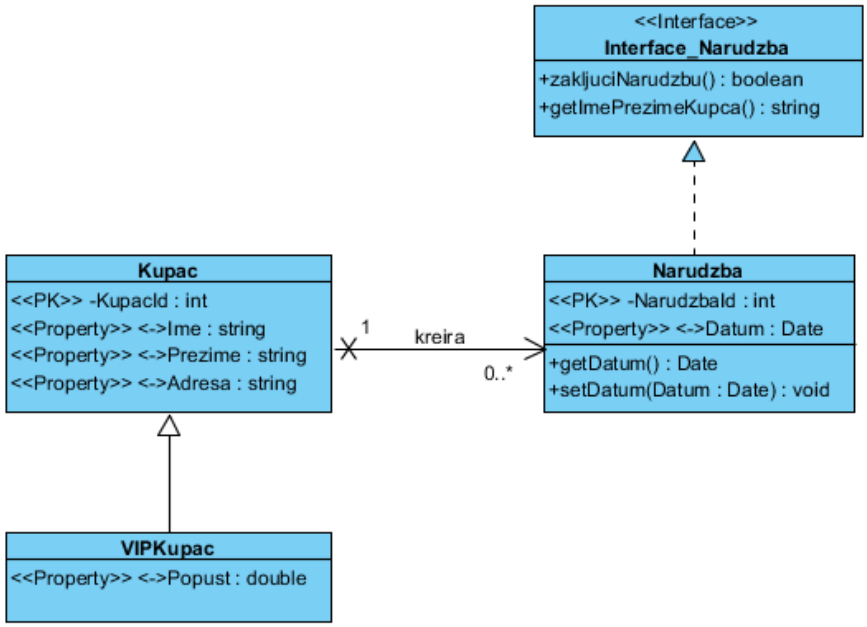
```
private Narudzba[] _Narudzba
```

Atribut *_Narudzba* je niz objekata tipa Narudzba kako bi bilo moguće pohraniti kolekciju narudžbi za jednog kupca.

1.7. RELACIJE NASLJEĐIVANJA

Klasa može naslijediti više osobina (atributa) od svojih roditelja putem veze generalizacije.

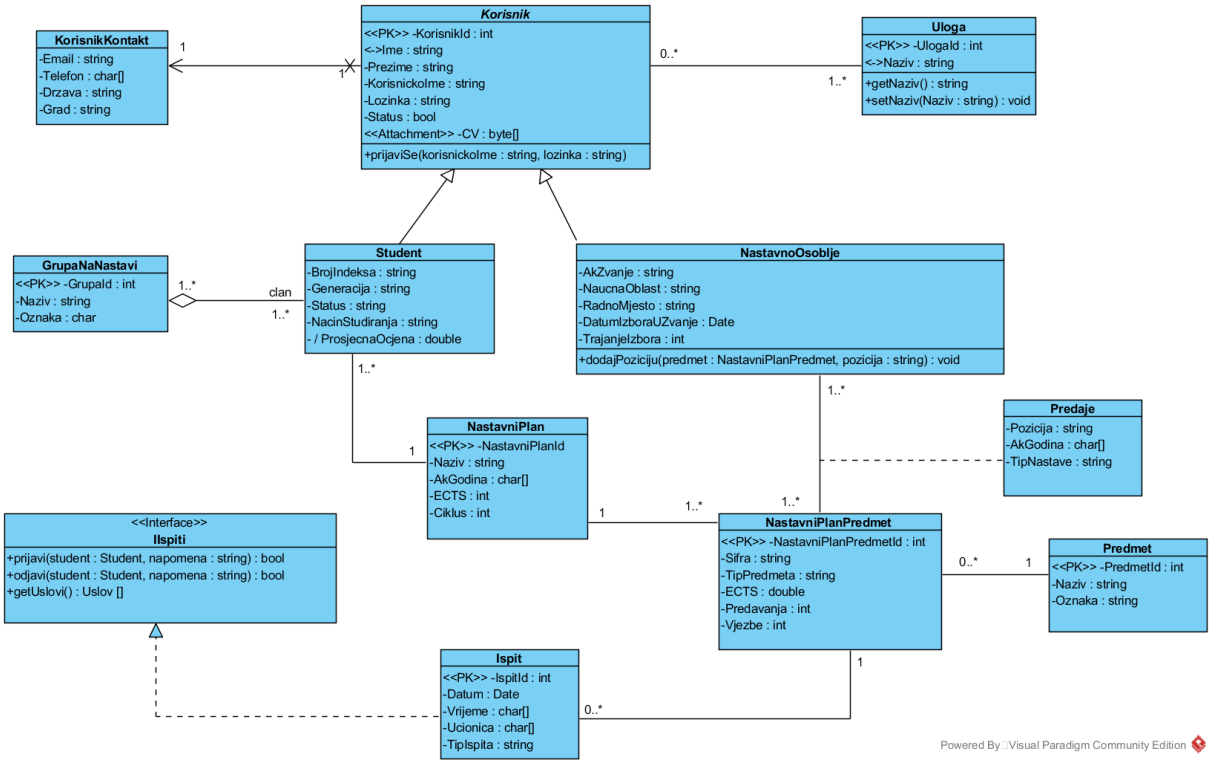
Primjer:



Podklasa VIPKupac nasljeđuje asocijaciju nadređene klase („*kreira*") Narudzba i sve njene atribute i operacije.

2. Zadaci

U nastavku je prikazan dijagram klasa koji obuhvata jedan segment projekta eUniverzitet, sa akcentom na specifične tipove relacija.



Zadatak za vježbu:

Rent-a-car agencija:

Klijent se, kada želi iznajmiti automobil, javlja *referentu* koji je zadužen za iznajmljivanje. Klijent popunjava zahtjev za iznajmljivanje u koji unosi osnovne podatke o sebi i bira auto koje želi iznajmiti i na koji period.

- *Referent* pregleda zahtjev i upoznaje klijenta sa pravilima iznajmljivanja automobila te priprema ugovor o iznajmljivanju. Referent i klijent potpisuju ugovor i klijent dobija ključeve i dokumente od automobila.
- Ako ima problema sa automobilom klijent kontaktira agenciju i dobiva drugi automobil na korištenje ili *serviser* popravlja auto na licu mjesta ako je moguće. Serviser pravi zapisnik o servisu. Ukoliko je kvar na autu izazvan krivicom klijenta, sačinjava zapisnik sa procjenom štete koja se naplaćuje od klijenta.
- Nakon isteka ugovora o iznajmljivanju klijent vraća auto agenciji. Prilikom vraćanja auta klijent dobiva obračun (fakturu) za iznajmljivanje automobila i eventualne štete.
- Klijent plaća fakturu gotovinom ili karticom.

a) Izraditi **dijagram klasa** za opisani sistem.
b) Modelirati funkcionalnost sistema use-case dijagramom.

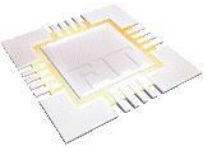
Ispitni zadatak:

Turistička agencija je izdala specifikaciju zahtjeva za razvoj softvera za upravljanje osnovnim poslovnim aktivnostima i bolju organizaciju turističkih grupa i putovanja. Prepoznate su 3 osnovne grupe korisnika: voditelj agencije, turistički vodiči i turisti.

- Voditelj turističke agencije unosi podatke o turističkim vodičima i njihovim zaduženjima, te podatke o aktuelnim turističkim putovanjima (*Lokacija, VaziOd, VaziDo, Polazak, Trajanje, Opis, Cjenovnik*). Prilikom evidencije putovanja potrebno je definisati i podatke o smještaju (*NazivHotela, Adresa, BrojZvjezdica*). Pri tome voditelj u svakom momentu može izmijeniti uslove ponude za putovanje. Svakom dodjelom zaduženja sistem turističkim vodičima šalje notifikaciju.

- Turistički vodiči mogu pregledati sva zaduženja, odnosno destinacije na kojima su dodijeljeni uz detalje o grupi prijavljenih turista. Pored toga, oni imaju uvid u vlastitu statistiku koja uključuje pretragu svih proteklih zaduženja i pregled dobivenih ocjena od strane turista u odgovarajućim grupama.

- Turisti (*Ime, Prezime, DatumRodjenja, Spol, Email, Telefon*) prvenstveno mogu pretraživati ponude agencije (putovanja), kreirati rezervaciju uz mogućnost odabira i smještaja, te obaviti i online plaćanje. U slučaju da je popunjen definisani broj mjesta za odabranu ponudu, rezervacija će za turiste biti nedostupna. Prilikom uspješnog kreiranja rezervacije turistima se automatski dodjeljuje grupa kojoj pripadaju (svako putovanje se organizuje u 3 grupe od po 30 turista), te im se na mail šalje raspored aktivnosti za dodijeljenu grupu. Po završetku putovanja turisti imaju mogućnost ocjene vodiča. Rezervaciju je također moguće i otkazati najkasnije 10 dana prije putovanja. Sistem potom automatski definiše popust od 10% za sva preostala slobodna mjesta i šalje

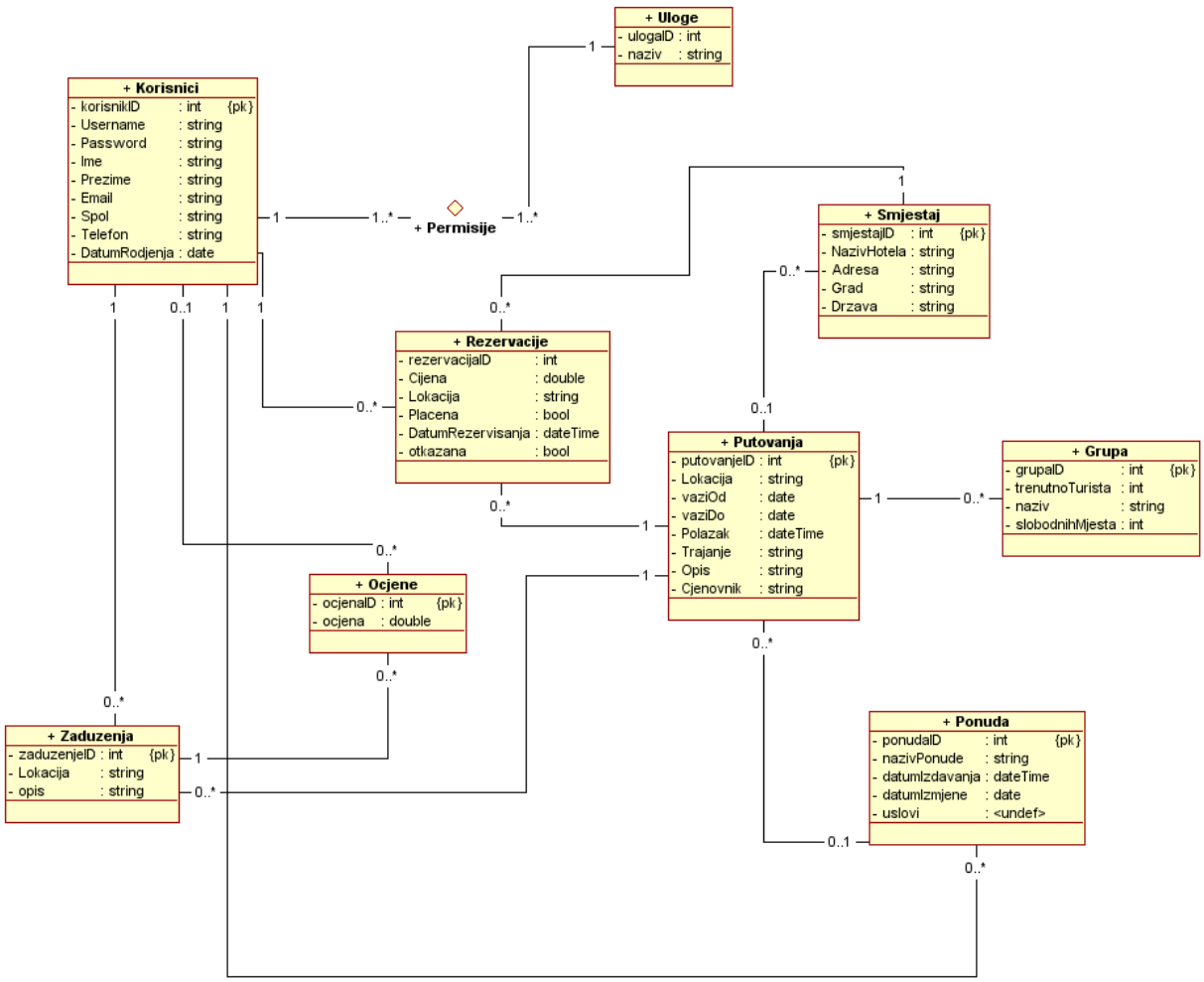


obavještenje putem mail-a svim registrovanim turistima agencije koji ranije nisu bukirali putovanje.

Izraditi dijagram klasa za dati sistem uz obavezno prepoznavanje specifičnih tipova relacija.

Studentsko rješenje:

Ispod je prijedlog rješenja studenta koji je osvojio 25 od mogućih 30 bodova. Dijagram je kreiran u alatu Open ModelSphere.



Prateće video lekcije možete pogledati u sklopu YouTube kanala na sljedećim linkovima:

- Dijagram klasa (Visual Paradigm 14.0)
- Dijagram klasa (Open ModelSphere 3.2)

Literatura

1. Martin Fowler, UML Distilled Third Edition: A brief guide to the standard object modeling language, 2004. Download (Dostupno: 06.03.2017.).
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/