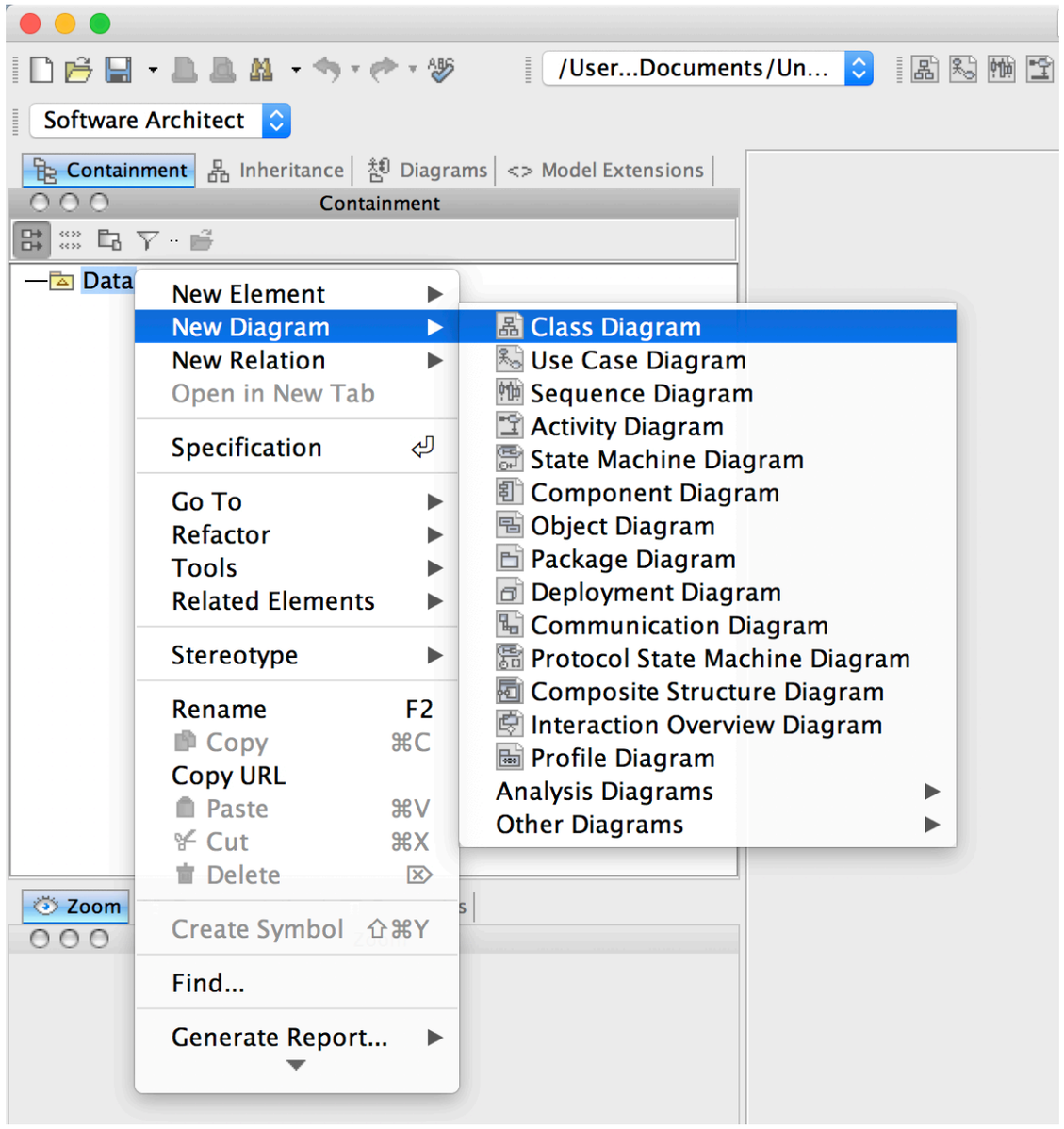
SIMS - VEŽBE 04

DIJAGRAMI KLASA

OSNOVNI KONCEPTI

- Dijagrami klasa spadaju pod dijagrame za opis strukture
- Dijagrami klasa se mogu koristiti tokom čitavog životnog ciklusa softverskog proizvoda
- U zavisnosti od količine detalja dele se na:
  - Konceptualne (domenske) modele - manje detaljni, fokusiraju se na modelovanje koncepta domena
  - Implementacione modele - mapiraju se direktno na programski kod
- Link ka delu dokumentacije Magic Draw-a koji se tiče dijagrama klasa.

KREIRANJE NOVOG DIJARAMA KLASA

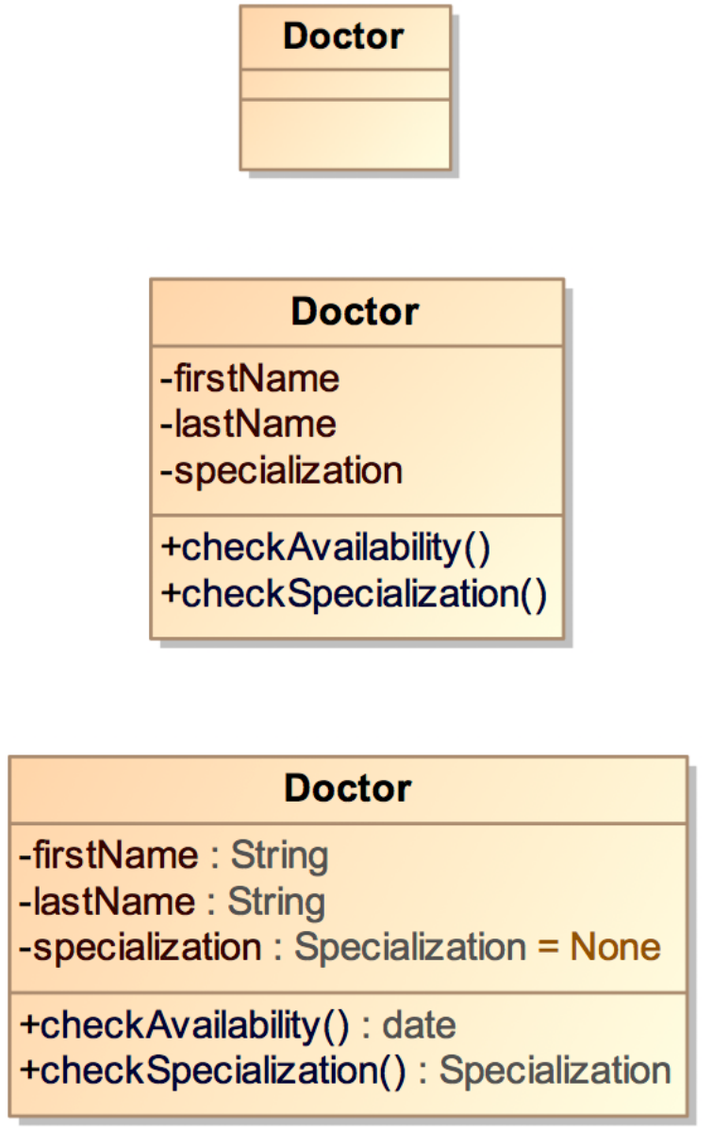


OSNOVNI ELEMENTI DIJAGRAMA KLASA

- Dijagrami klasa obično sadrže:
  1. Klase
     - Obeležja
     - Metode
  2. Interfejse
  3. Veze
     - Asocijacije
     - Generalizacije
     - Veze zavisnosti
     - Implementacije interfejsa

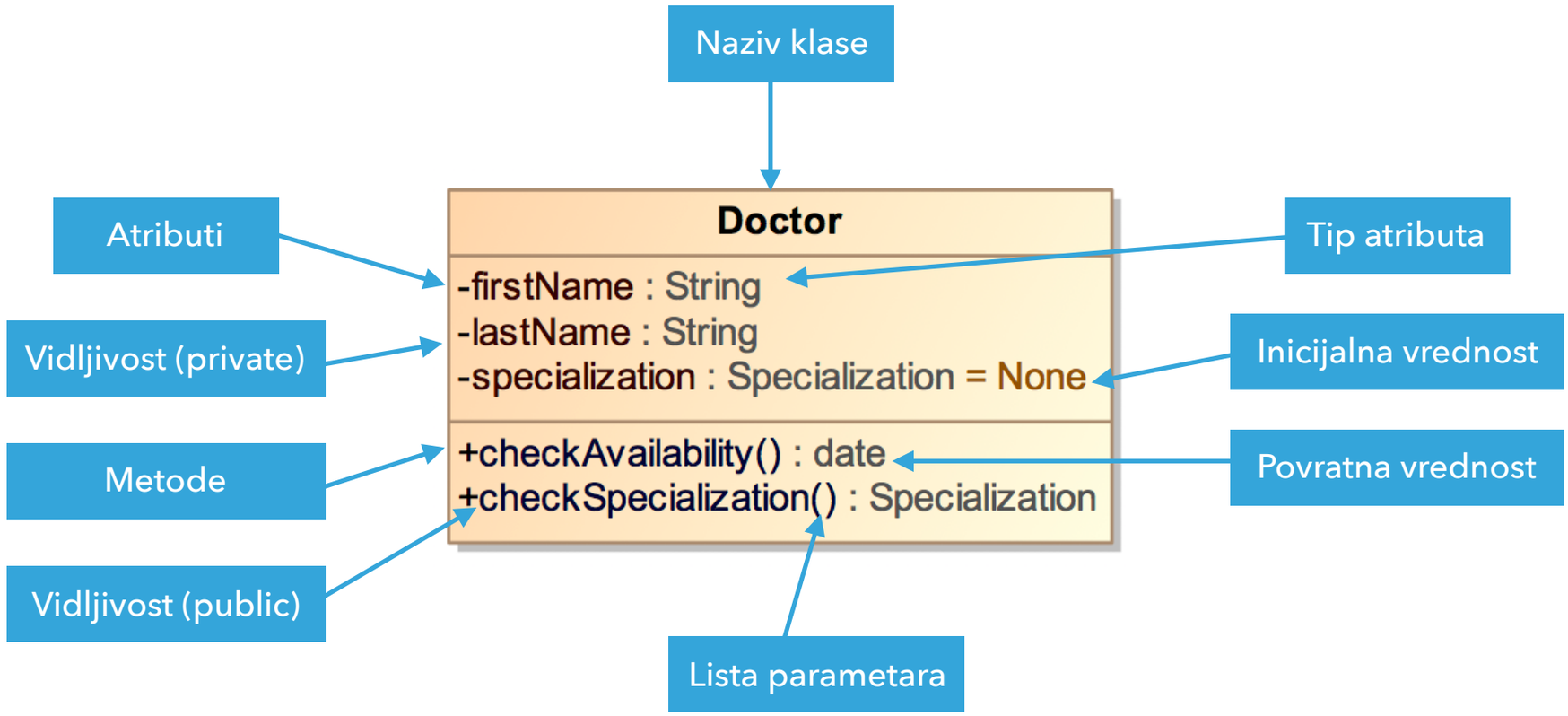
KLASE

- Klase predstavljaju šablone na osnovu kojih se kreiraju objekti koji su relevantni za neki sistem
- Klasama možemo opisivati osobe (npr. doktori, pacijenti), stvari (npr. bolnica, operaciona sala), događaji (npr. operacija) ili apstraktni neki koncepti
- U objektno orijentisanim jezicima, poput Jave, programi se kreiraju na osnovu klasa
- Relevantne osobine svake klase predstavljaju se pomoću njenih strukturalnih karakteristika (obeležja tj. atributa) i njenog ponašanja (metoda)



Nivoi detalja u dijagramima klasa

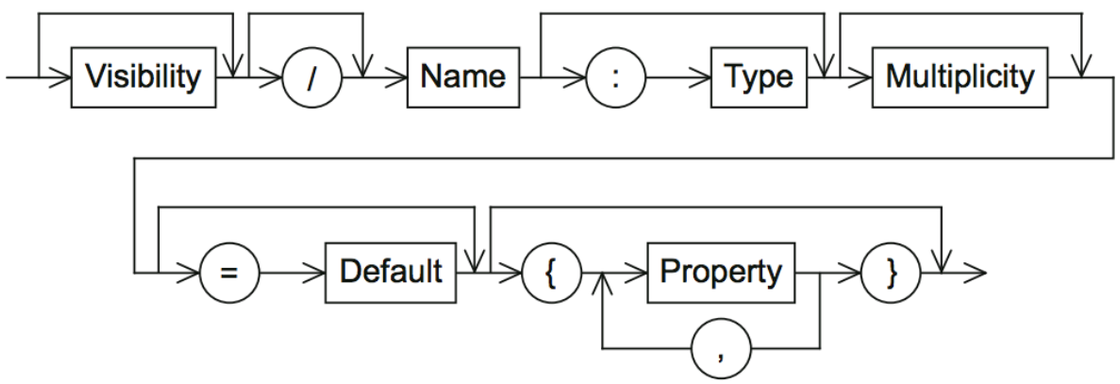
KLASE


Naziv klase
Doctor
-firstName : String
-lastName : String
-specialization : Specialization = None
+checkAvailability() : date
+checkSpecialization() : Specialization
Atributi
Tip atributa
Vidljivost (private)
Inicijalna vrednost
Metode
Povratna vrednost
Vidljivost (public)
Lista parametara

OBELEŽJA KLASE

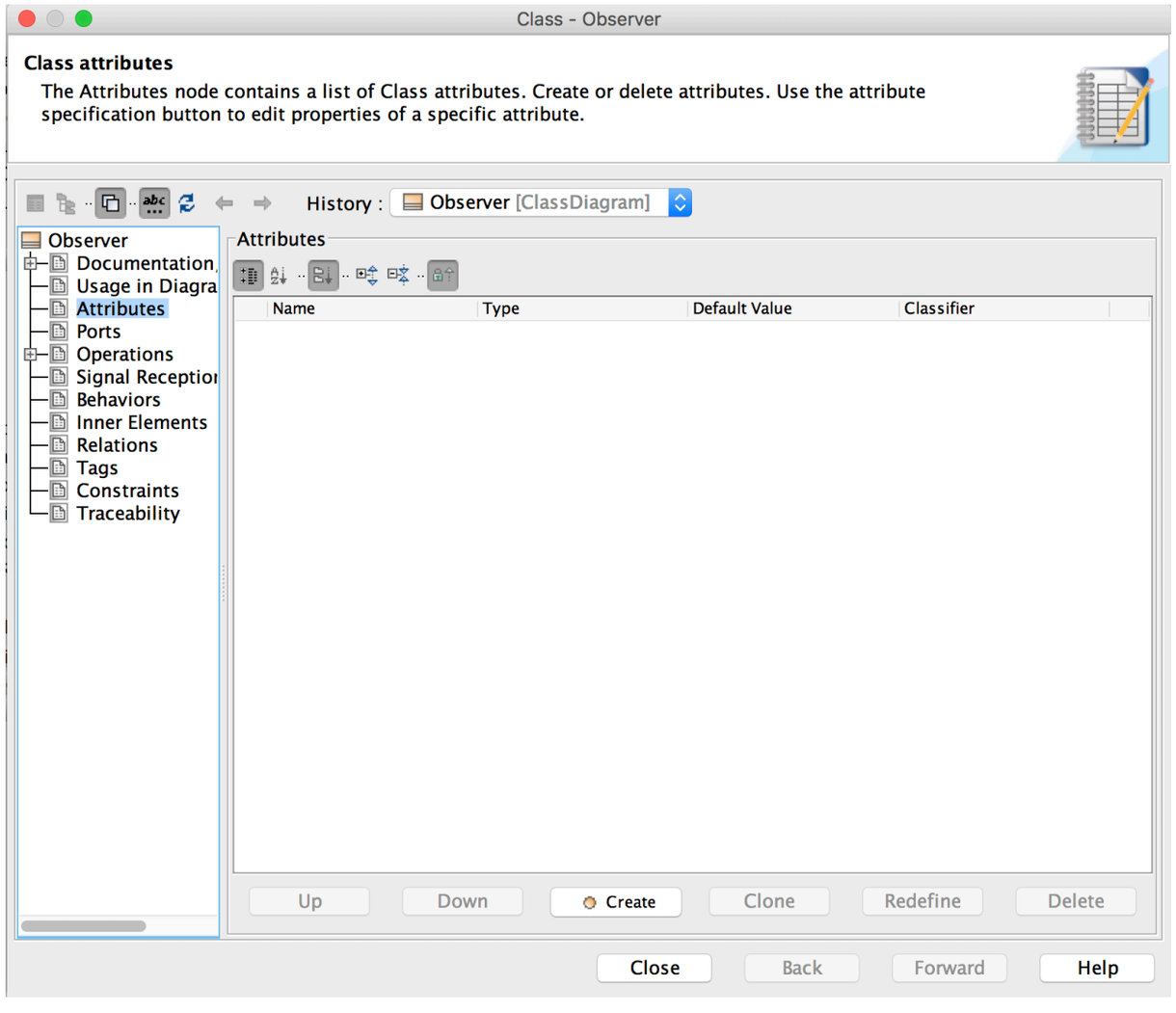
- Za imenovanje obeležja koristi se *CamelCase* notacija
- Osnovne osobine obeležja su:

1. Vidljivost (*visibility*)
2. Naziv (*name*)
   - jedino obavezno polje
3. Tip podatka (*type*)
4. Pozdrazumevana vrednost (*default*)
5. Kardinalitet (*multiplicity*)



Format za specifikaciju obeležja

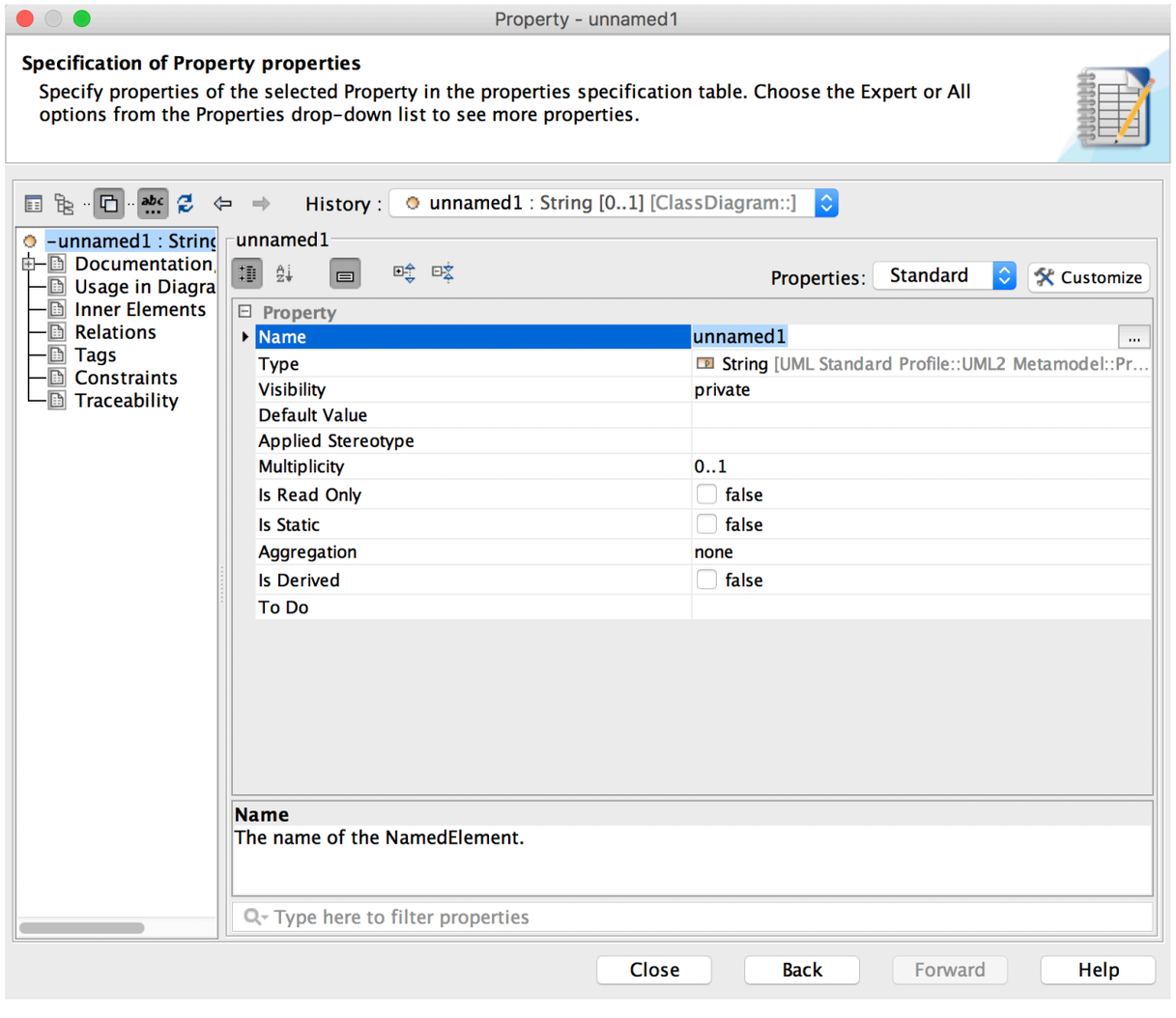
DODAVANJE NOVOG OBELEŽJA



- Dodavanje novog obeležja se vrši dvoklikom na željenu klasu i odabirom Attributes kartice, nakon čega se otvara prikazani dijalog

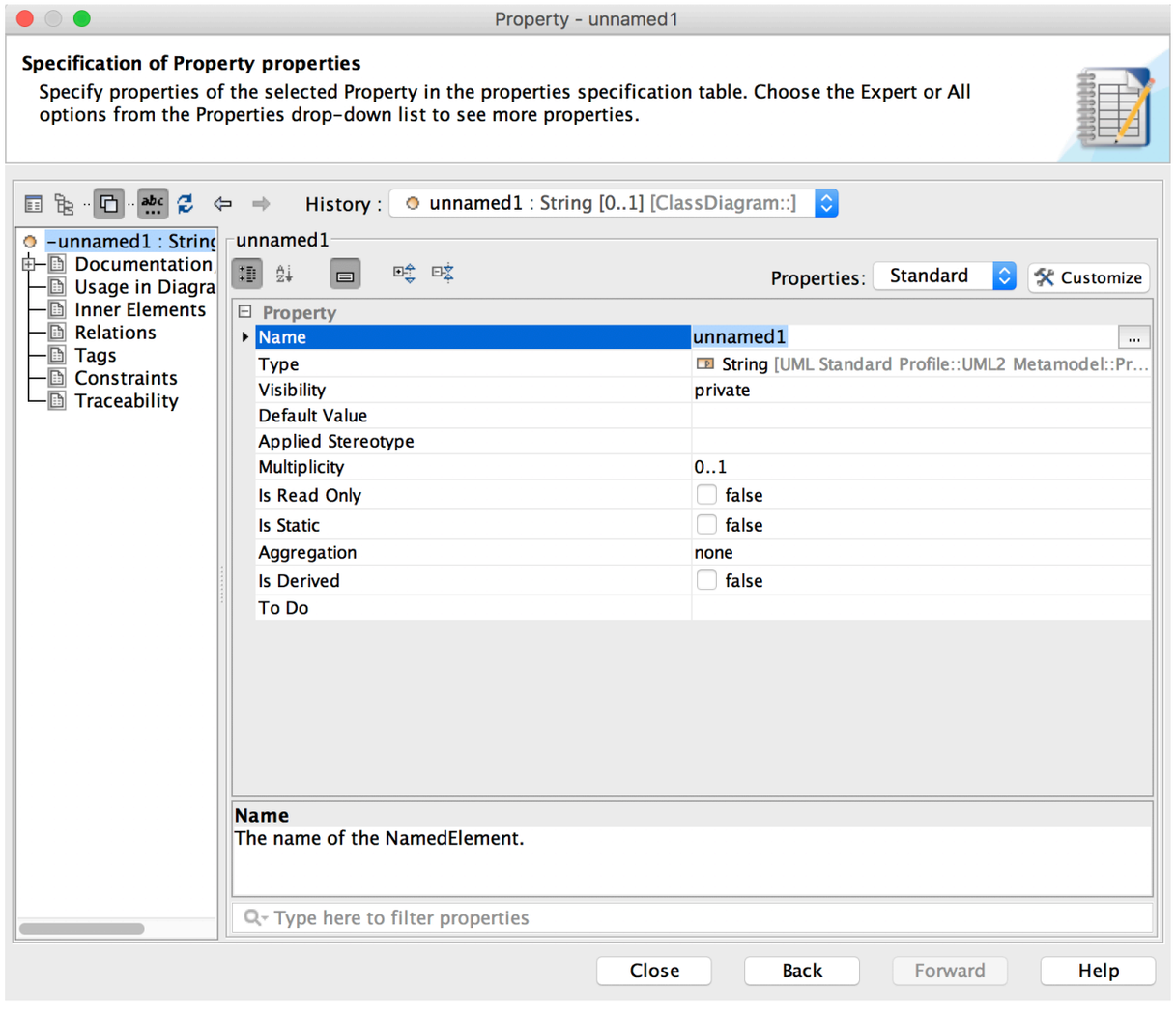
- Ukoliko klasa nema definisanih obeležja, potrebno je kliknuti na dugme Create kako bi se otvorio dijalog za manipulaciju nad obeležjem

DODAVANJE NOVOG OBELEŽJA



- Name - desktiptivan naziv obeležja pisan u Camel Case notaciji
- Type - tip može biti neki od unapred definisanih vrednosti, enumeracija, klasa ili interface
- Visibility - vidljivost obeležja može biti:
    1. Private **-** obeležje je vidljivo samo unutar klase u kojoj je definisano, tj. druge klase im ne mogu pristupati
    2. Public **+** obeležje uz koje stoji je javno, dakle, može mu se pristupati iz bilo koje klase
    3. Protected **#** obeležje je vidljivo samo unutar klase u kojoj je definisano i potklasama koje su izvedene (nasleđene) iz te klase
    4. Package **~** obeležje je vidljivo svim klasama koje se nalaze unutar istog paketa kao i klasa unutar kog je definisano
- Default value - inicijalna vrednost koju obeležje dobija prilikom instanciranja klase

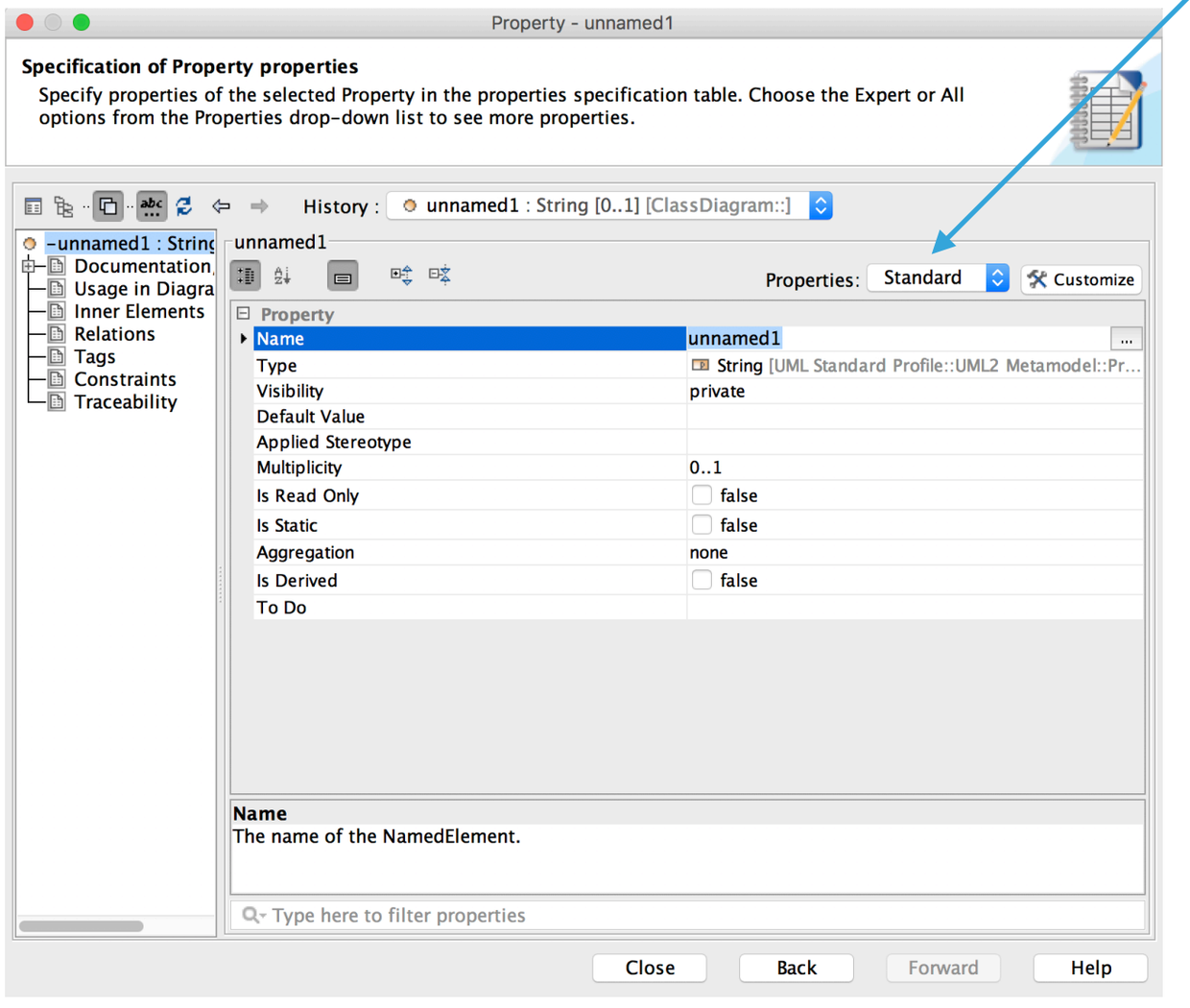
DODAVANJE NOVOG OBELEŽJA



- Applied Stereotype- mogućnost dodavanja predefinisanih i korisničkih sterotipa, npr. Moguće je uvesti stereotip <<const>> čime bismo označili da je obeležje konstanta
- Multiplicity - kardinalitet označava opseg broja vrednosti koje objekat posmatrane klase može da ima za dati atribut
  - 0 obeležje ne sme imati vrednost
  - 0..1 obeležje može, ali i ne mora imati vrednost
  - 0..* obeležje može da nema vrednost, ili može imati neograničeno mnogo vrenosti
  - 1 podrazumeva vrednost, obeležje sadrži tačno jednu vrednost koja ne sme biti nedefinisana
  - 1..* obeležje ima tačno jednu ili neograničeno mnogo vrenosti
  - * proizvoljan broj vrednosti obeležja

DODAVANJE NOVOG OBELEŽJA

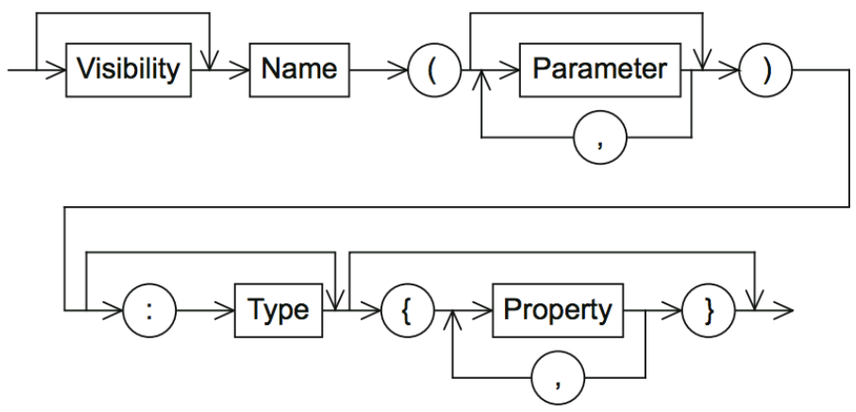
Odabirom Expert pogleda, dobijate mogućnost rada sa dodatnim setom property-ja



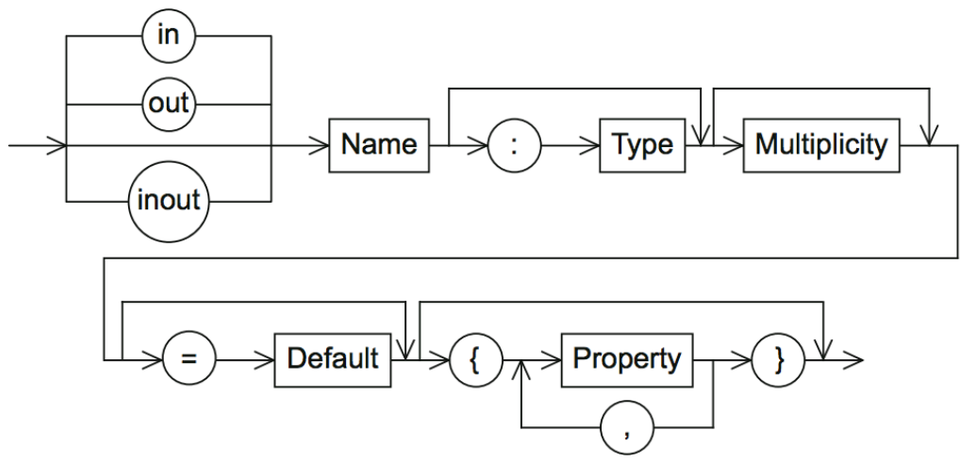
- Is Read Only - ne može se menjati nakon inicijalizacije, zabranjeno je posedovanje set metode
- Is Static - statičko obeležje je vezano za klasu, a ne za instancu klase (podvučeno)
- Aggregation - speficira se tip agregacije
  - None
  - Shared
  - Composite
- Is Derived - vrednost atributa može da se izvede iz vrednosti ostalih atributa

METODE KLASA

- Za imenovanje metoda koristi se *CamelCase* notacija
  - *bookConsultation, getDoctor, checkSpecialization*

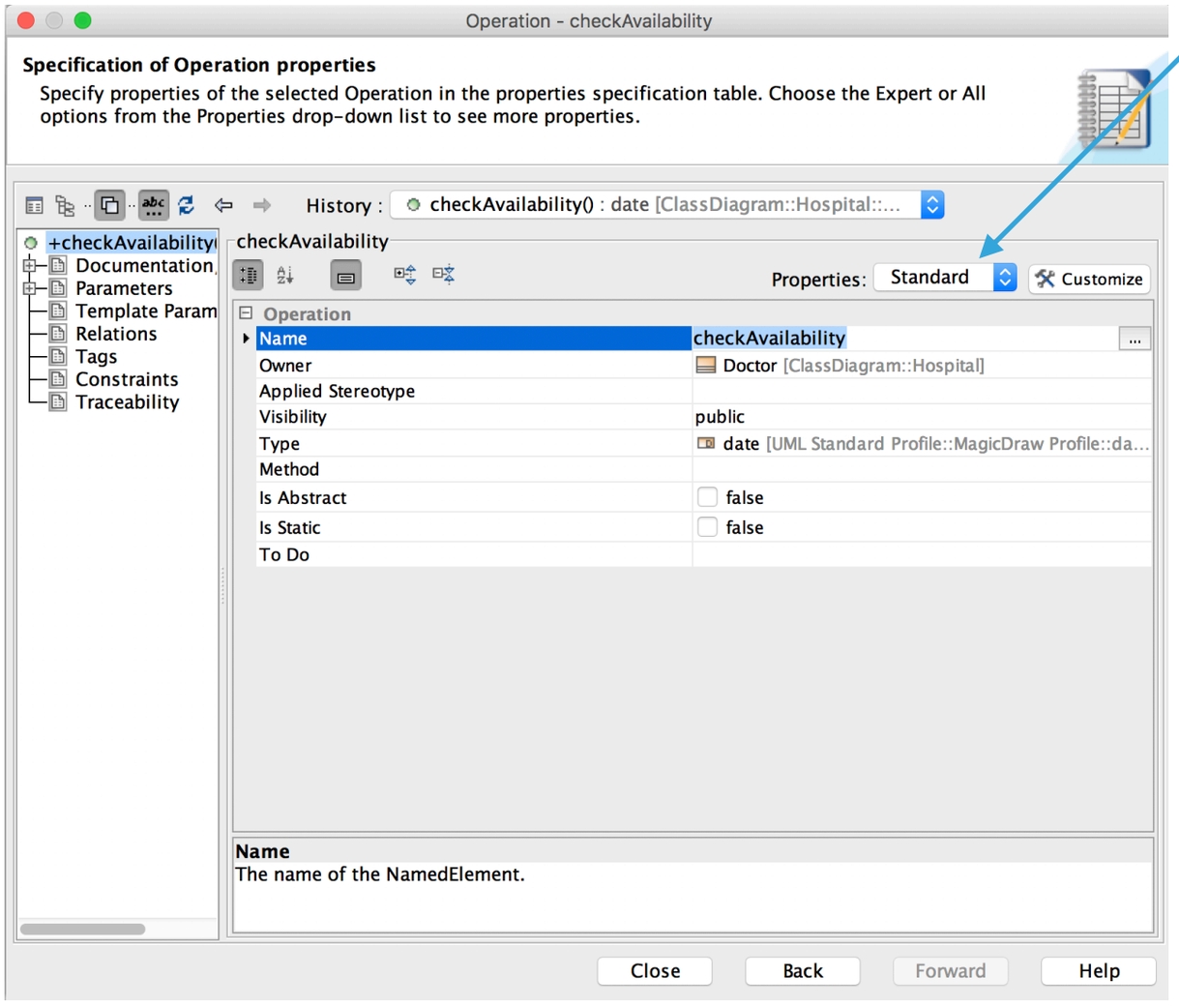


Format za specifikaciju metoda



Format za specifikaciju parametara metode

DODAVANJE NOVE METODE



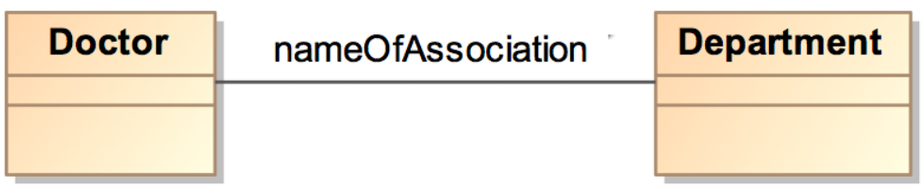
Odabirom Expert pogleda, dobijate mogućnost rada sa dodatnim setom property-ja

- Dodavanje nove metode se vrši dvoklikom na željeljenu klasu i odabirom Operations kartice. Klikom na dugme Create otvara se prikazani dijalog
- Name - desktiptivan naziv metode pisan u Camel Case notaciji
- Visibility - vidljivost ista kao i kod obeležja
- Type - povratna vrednost može biti neki od unapred definisanih vrednosti, enumeracija, klasa ili interface
- Is Abstract - apstraktne metode mogu biti deo samo apstraktnih klasa, kao takve ne mogu imati telo. Implementacija mora biti definisana od strane klasa naslednica
- Is Static - statičke metode su vezano za klasu, a ne za instancu klase (podvučeno)

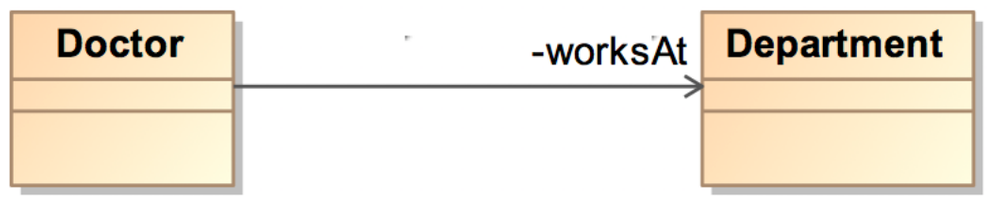
Asocijacija modeluje “uses” vezu. Doktor koristi Departman, ali i Departman koristi Doktora

VEZA ASOCIJACIJE

- Asocijacijom se modeluju veze klasa (kažemo - klasa A referencira klasu B)
- Veza asocijacije može biti:

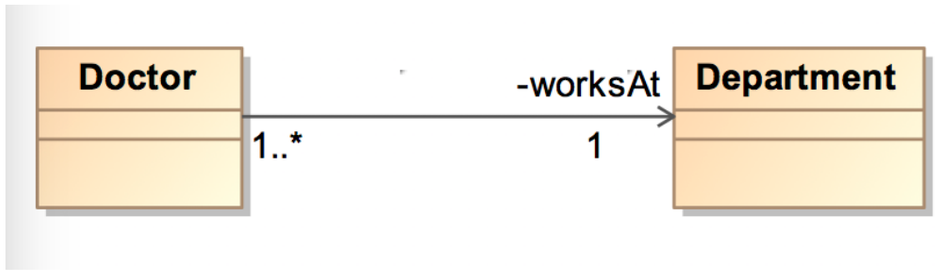


Navigabilna u oba smera (biderikciona)



Navigabilna u jednom smeru

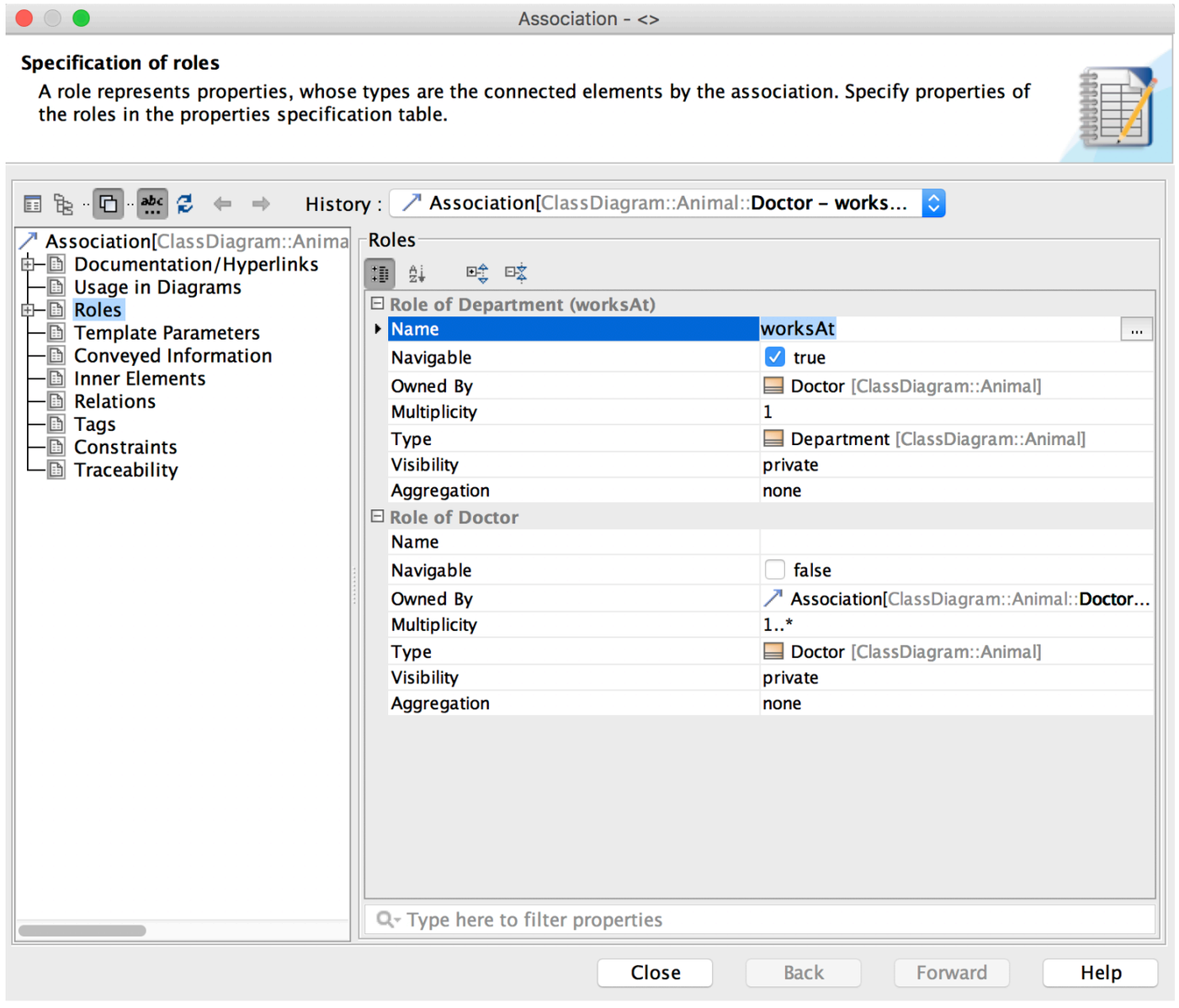
- Veza asocijacije opciono može imati naziv, koji se piše po pravilima za imenovanje klasa
- Na osnovu naziva uloge se imenuju obeležja. Naziv obeležja odgovara nazivu uloge suprotnog kraja asocijacije. Nazivi uloga se pišu po pravilima za imenovanje obeležja
- U vezi asocijacije, relacija između dve klase može biti definisana i informacijom o broju objekata (instanci) sa druge strane veza. Kardinalitetom veze se određuju broj objekata koji mogu biti povezan sa tačno jednim objektom suprotne strane



Doktor radi na tačno jednom departmanu, dok departman ima jednog ili više doktora

VEZA ASOCIJACIJE

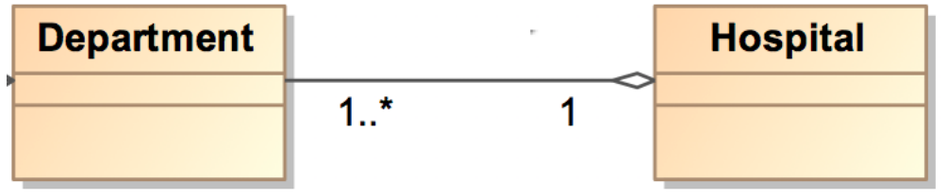
- *Name* - opis uloge klase u okviru veze (pretvara se u ime atributa u suprotnoj klasi). Ako se ne definiše, za ime atributa se koristi ime klase
- *Navigable* - oznaka da li instanca jedne klase može da pristupi instanci druge klase
- Visibility - pošto se veza asocijacije prilikom generisanja koda prevodi u atribut jedne ili obe klase (u zavisnosti od kardinalnosti i usmerenosti veze), Visibility definiše nivo pristupa tom atributu: private, protected, public ili package
- Aggregation - definiše tip asocijacije
  - none - veza asocijacije
  - shared - veza agregacije
  - composite - veza kompozicije



Agrergacija modeluje “has-a” vezu. Bolnica se sastoji od odeljenja

VEZE AGREGACIJE I KOMPOZICIJE

- Agregacija je vrsta asocijacije kojom se modeluje odnos “celina-deo”, pri čemu su delovi samostalne klase čije instance mogu da “žive” i nezavisno od klase koja ih agregira
- Romb se stavlja kod klase koja predstavlja celinu



- Kompozicija je specijalna vrsta agregacije kod koje su delovi slabi objekti koji ne mogu da samostalno postoje bez osnovne klase i njihov životni ciklus je čvrsto povezan
- Kada se briše celina, brišu se I njegovi delovi
- VAŽNO: Kardinalitet na strani koja predstavlja celinu je uvek 1

ASOCIJATIVNE KLASE

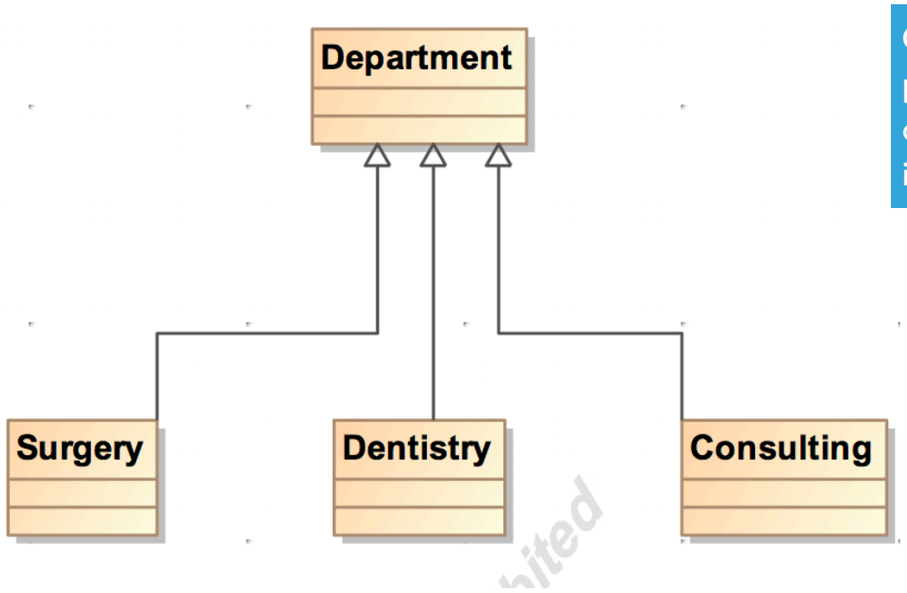
- Ukoliko je potrebni dodelite atribute ili metode u odnos između jedne ili više klasa, a ne samoj klasi, to se može učiniti pomoću asocijativne klase
- Asocijativna klasa sadrži dodatne osobine asocijacije između dve klase

Generalizacije modeluje “is-a” vezu. Hirurgija, stomatologija su odeljenja bolnice

VEZA GENERALIZACIJE

- Vezom generalizacije definišemo da se atributi, operacije i odnosi klase višeg nivoa (superklasa) nasleđuju, tj.postaju dostupni klasi nižeg nivoa (podklasa)
- Traženje zajedništva u atributima i operacijama grupe klasa i njihovo postavljanje u zajedničku klasu naziva se *generalizacija* (zahteva domensko znanje i apstraktno razmišljanje)
- Izvođenje klasa na osnovu postojećih u posebne klase naziva se *specijalizacija* (zahteva konkretno razmišljanje, pogled ka implementaciji)
- Klase nasledice pored toga što nasleđuju deo ponašanja od pretka, mogu da dodaju i prošire svoje ponašanje dodavanjem novih atributa i metoda

Generalizacija je jaka zavisnost, potomak mora preuzeti sve osobine pretka - ne može izostaviti one koje ne želi



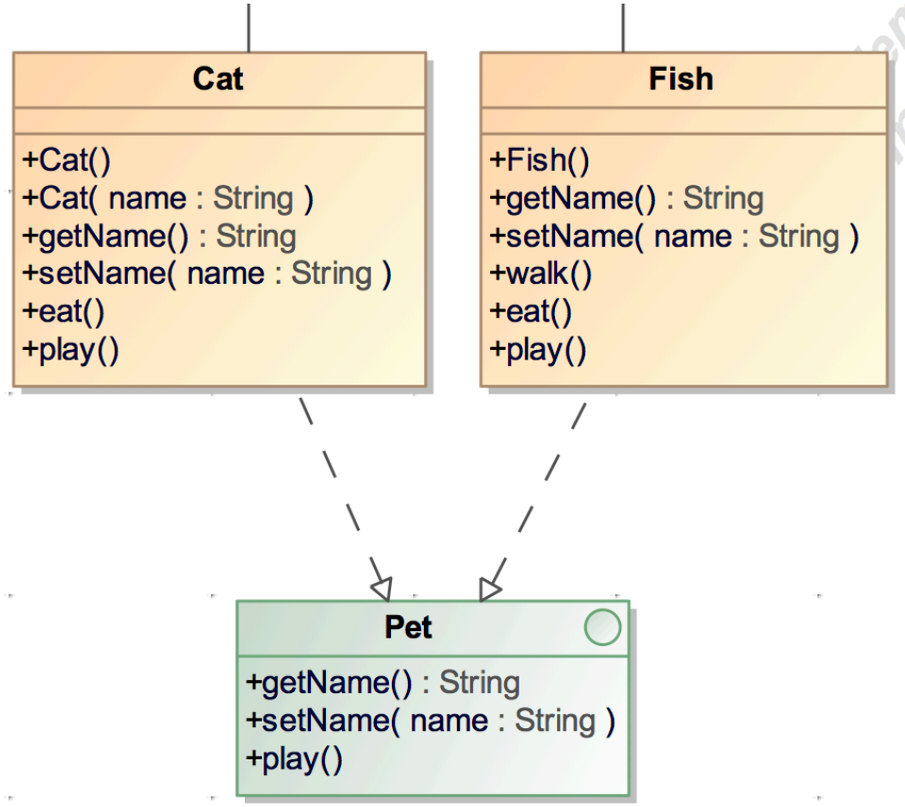
Smer generalizacije je od naslednika ka pretku

VEZA ZAVISNOSTI

- Veza zavisnosti (*dependency*) predstavlja vezu između dva elementa modela kojom se definiše da funkcionalnost ili implementacija jednog elementa modela zahteva prisustvo drugog elementa modela
- Kao element modela na dijagramu klasa može se posmatrati klasa, interfejs ili paket
- Element modela od koga polazi usmerena linija predstavlja zavisan element, dok element na drugom kraju veze predstavlja nezavisan element. Ovom vezom se definiše da izmena u nezavisnom elementu modela utiče na zavisan element modela

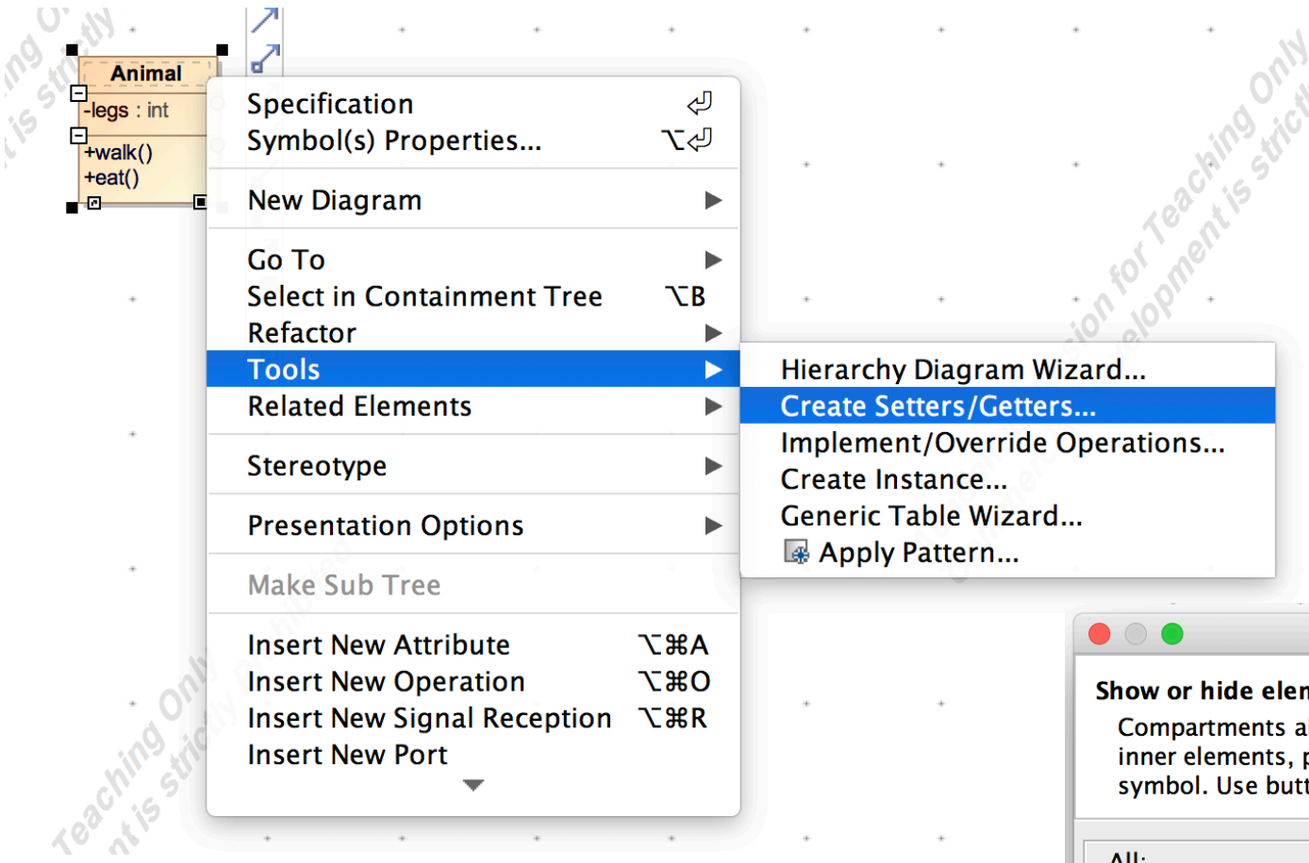
VEZA IMPLEMENTACIJE INTERFEJSA

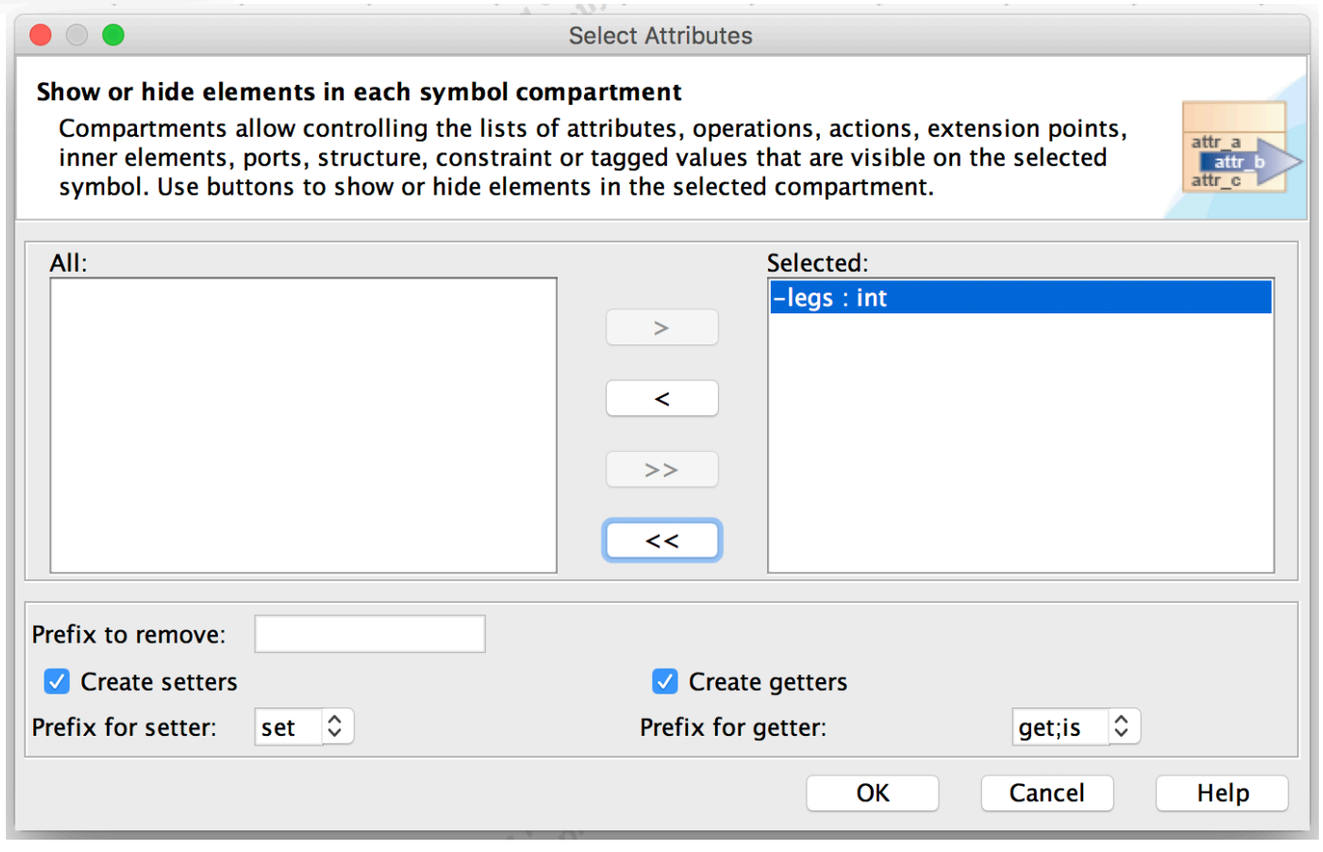
- Interface predstavlja spisak metoda, pri čemu svaka klasa koja ga implementira mora obezbedi telo ovih metoda
- Klasa može da implementira više interfejsa, čime se dobija efekat višestrukog nasleđivanja



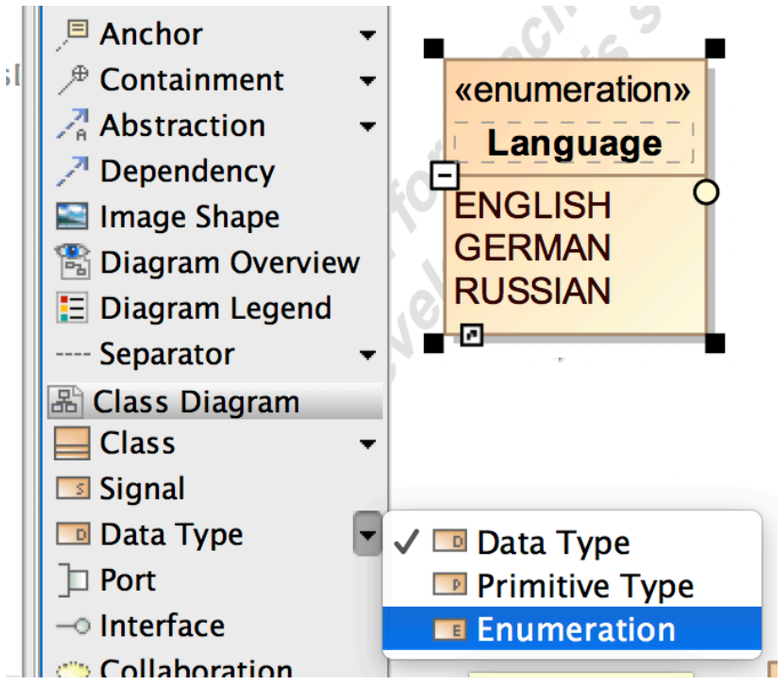
GENERISANJE GETTERA I SETTERA

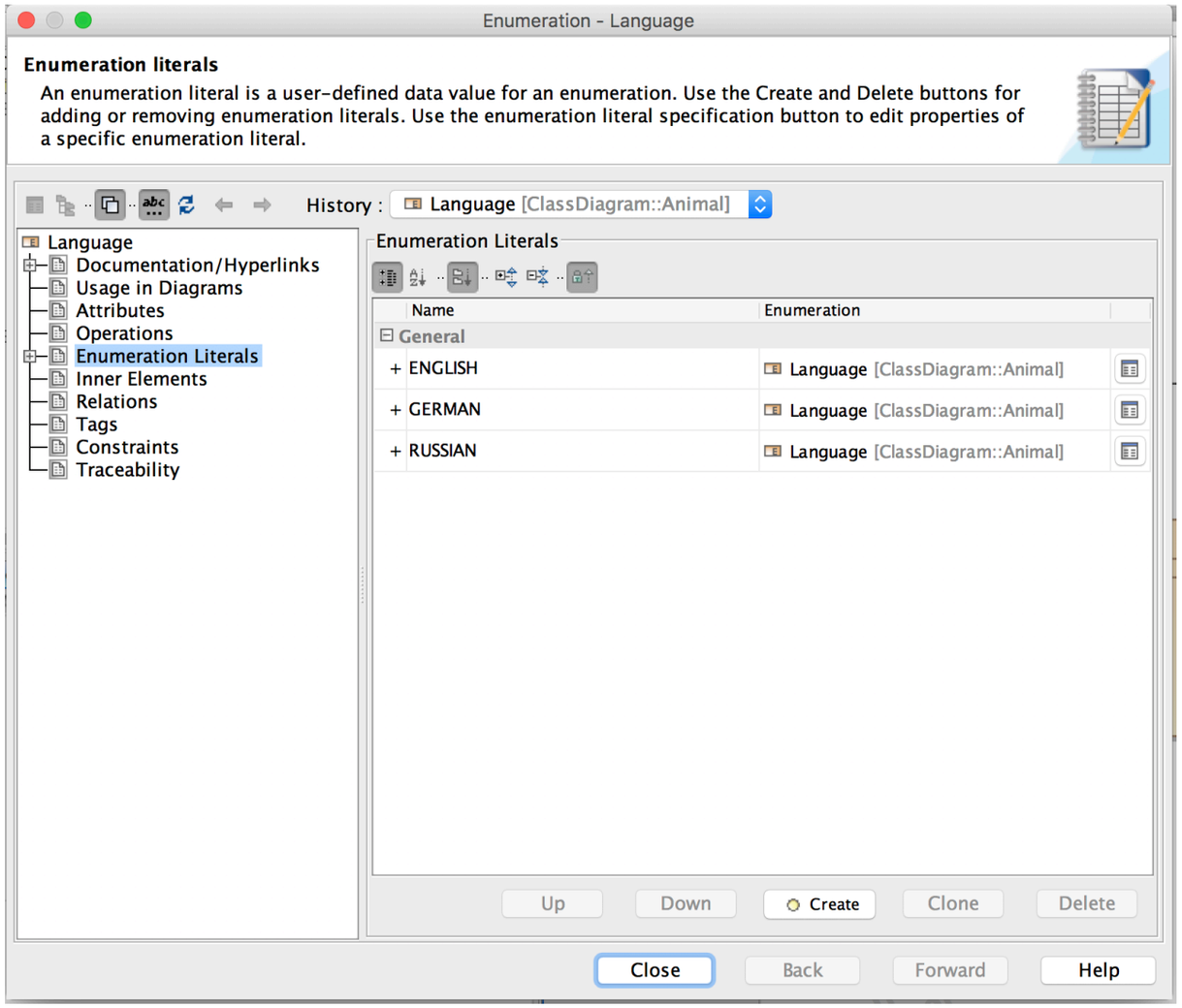
Desni klik na prethodno selektovanu klasu -> Tools -> Create Setters/Getters





KREIRANJE ENUMERACIJE





ZADATAK 1

Modelovati dijagram klasa za sledeći scenario.

- Referent kadrovske službe obavlja sledeće poslove:
    - Unos i izmenu podataka o novim zaposlenima (ime, prezime, adresa, pol, datum rođenja)
    - Raspored zasposlenog na radno mesto (naziv radnog mesta i broj bodova). Prilikom zasnivanja radnog odnosa, referent prvo unosi podatke o novom radniku i zatim ga raspoređuje na neko od postojećih radnih mesta.
    - Premeštanje zaposlenog na novo radno mesto.
    - Prekid radnog odnosa iz bilo kog razloga (odlazak u penziju, sporazumni raskid, otkaz).
    - Prikaz izveštaja o aktivnim radnicima.
    - Prikaz izveštaja o penzionerima.
    - Prikaz izveštaja o svim radnim mestima radnika – tekućem i nekadašnjim, sa datumom raspoređivanja na dato radno mesto.
- Administrator obavlja kreiranje korisničkih naloga za zaposlene. Promenu svoje lozinke mogu da obavljaju svi koji imaju korisnički nalog.
- Na početku svakog meseca aplikacija treba da proveri sve aktivne zaposlene i šefu kadrovske evidencije pošalje spisak onih koji u narednih mesec dana pune 65 godina (spisak kandidata za penziju).
- Šef kadrovske službe treba da ima mogućnost da radi sve poslove kao i referent, a dodatno i da održava podatke o raspoloživim radnim mestima.

ZADATAK 2

- Modelovati dijagram klasa za sledeći scenario.
  - Blog mogu da koriste registrovani i neregistrovani korisnici (bloggeri i svi ostali).
  - Samo registrovani korisnici mogu da pišu objave, jedan korisnik može imati više objava.
  - Objava pripada jednom korisniku.
  - Jedna objava može da ima više komentara, dok komentar pripada jednoj objavi.
  - Registrovani i neregistrovani korisnici mogu da ostavljaju komentare.
  - Objava može da pripada jednoj kategoriji, a ne mora nijednoj.
  - Kategorija može da ima više objava.
  - Postoji hijerahija kategorija:
    - Svaka kategorija može da ima jednu nadkategoriju i proizvoljan broj podkategorija

User/Korisnik:

- id: Integer
- name: String
- username: String
- password: String

Post/Objava:

- id: Integer
- title: String
- date: String
- content: String

Comment/Komentar:

- id: Integer
- title: String
- content: String
- date: String

Category/Kategorija:

- id: Integer
- name: String

ZADATAK 2/2

- Dodatna vežba:
  - Nakon modelovanja, izvršiti jednostavnu implementaciju aplikacije koja treba da podrži CRUD i testirati
  - Npr: Dodati kroz programski kod nekoliko korisnika, objava, komentara i kategorija I pokrenuti neku vrstu ispisivanja kreiranih elemenata.

ZADATAK 3

- Modelovati dijagram klasa za scenario Restorana