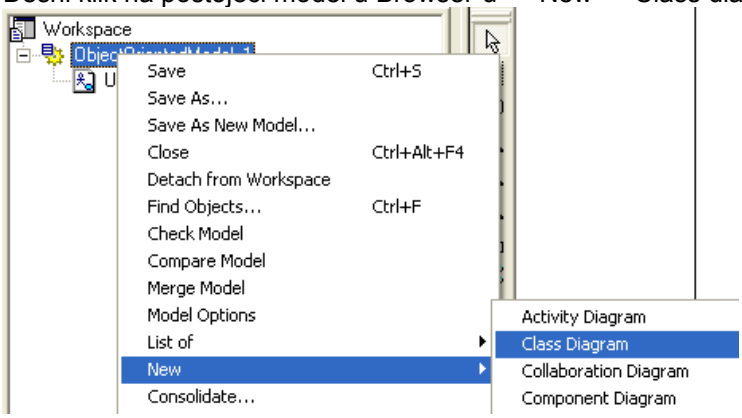
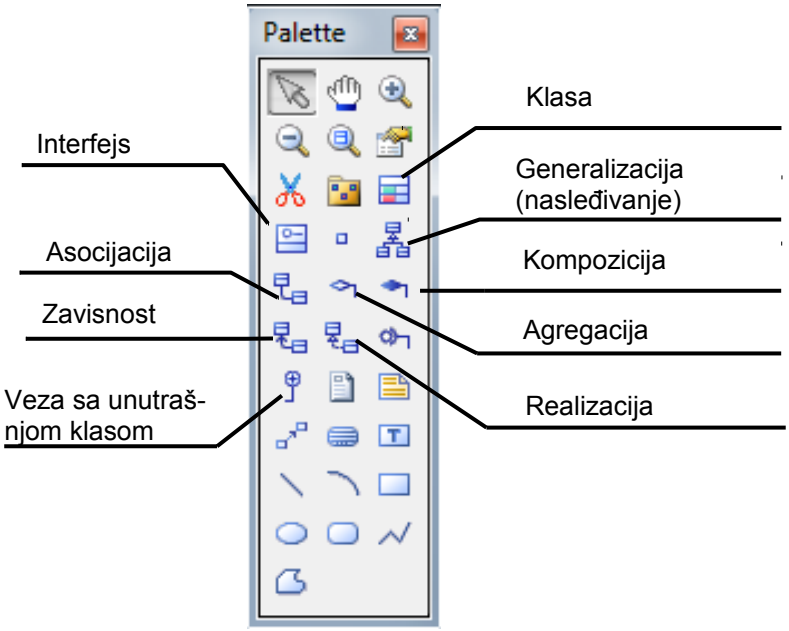
Uputstvo za kreiranje UML dijagrama klasa korišćenjem *PowerDesigner*-a 15.0

1. Kreiranje novog dijagrama klasa

Desni klik na postojeći model u Browser-u → New → Class diagram:



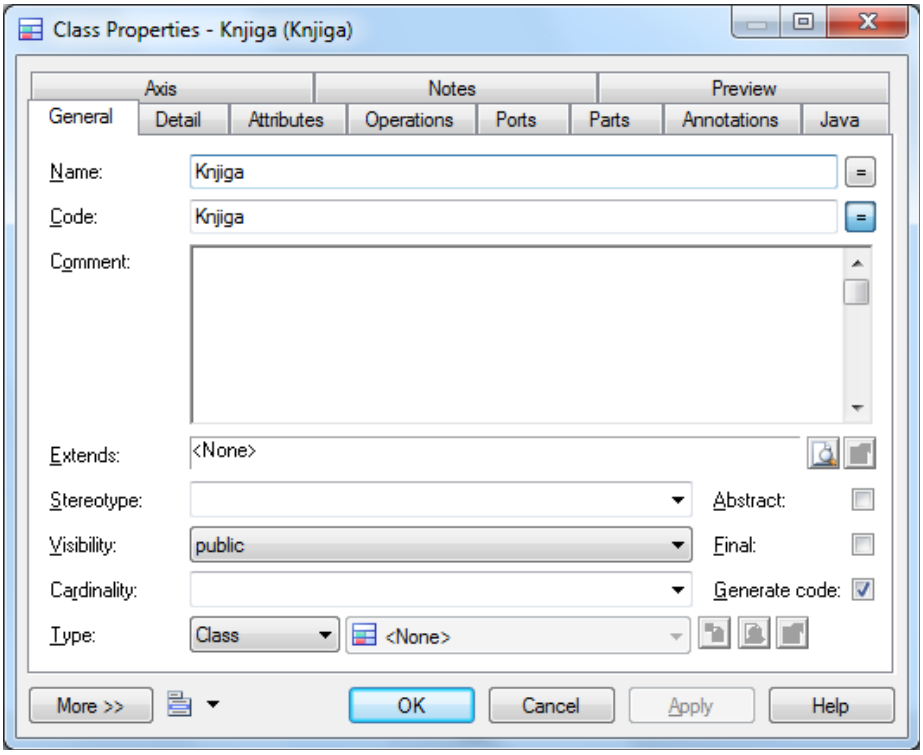
2. Korišćenje palete



3. Dijalozi za podešavanje osobina klase i interfejsa

U ovom odeljku opisan je način definisanja osobina klase, njenih atributa i operacija. U velikoj meri se ovaj opis odnosi i na interfejse, koji sadrže podskup ovde opisanih osobina klase.

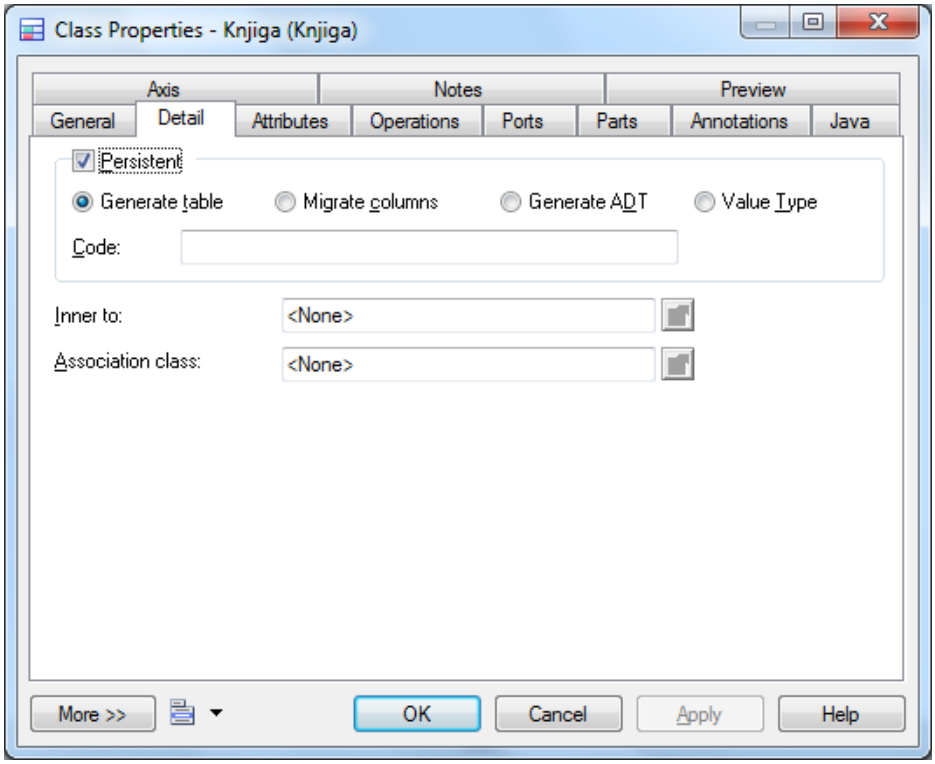
3.1 Kartica General



Najvažniji parametri:

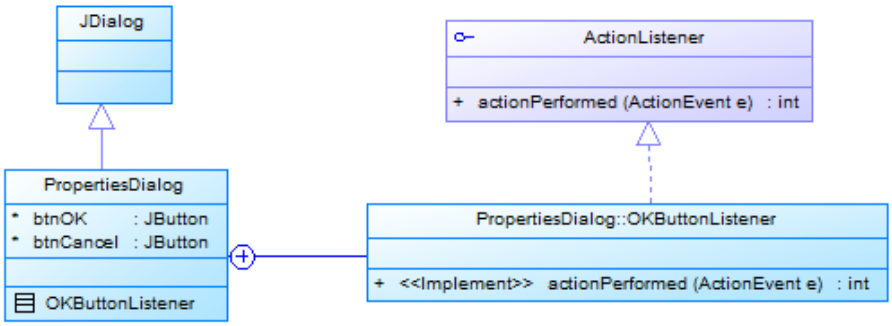
Name	Deskriptivan naziv klase
Code	Naziv klase koji se prenosi u programski kod
Comment	Opis klase
Extends	Predak date klase
Visibility	Vidljivost klase, označava kako je vide drugi objekti
Cardinality	Predviđeni broj instanci klase: 0..1, 1..1, 1..*, *
Type	Class – obična klasa Generic – generička klasa (v. odeljak 3.5 Generičke klase) Bound – konkretizacija generičke klase
Abstract	Oznaka da je klasa apstraktna
Final	Oznaka da je klasa poslednja u lancu nasleđivanja, odnosno da ne može imati naslednike
Generate code	Oznaka da je klasa predviđena za generisanje koda na izabranom programskom jeziku (uključuje se među objekte koji se generišu iz modela kada se pokrene proces generisanja)

3.2 Kartica Detail



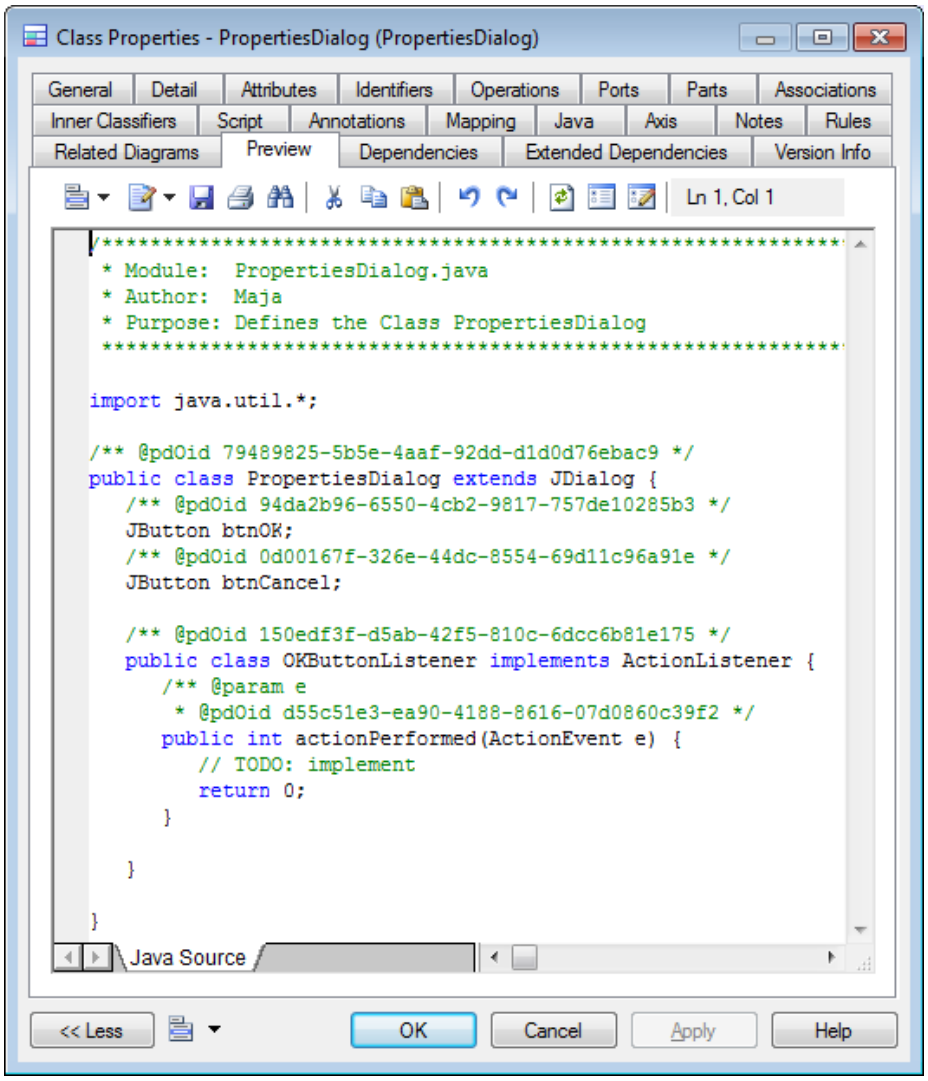
Persistent	Oznaka da je klasa perzistentna. Perzistentne klase su klase koje poseduju metode za smeštanje svojih atributa na disk i učitavanje sa diska, pri čemu "skladište" za smeštanje/učitavanje može biti datoteka ili baza podataka. U okviru *PowerDesigner*-a, ukoliko je klasa označena kao perzistentna, njeni podaci će biti preneti u fizički model baze podataka. Način na koji će ovaj proces biti obavljen definiše se podopcijama u okviru polja Persistent.
Inner to	Ime klase kojoj data klasa pripada, ukoliko se radi o unutrašnjoj klasi neke druge klase
Association class	Oznaka da klasa učestvuje u okviru veze asocijacije između druge dve klase (pogledati odeljak 4.1.3 Klasa asocijacije)

Unutrašnje klase se na dijagramu povezuju vezom "Inner Link". Na primer, na sledećoj slici je prikazan Java dijalog sa dva dugmeta i listenerom za dugme OK koji je implementiran kao unutrašnja klasa dijaloga:



3.3 Kartica Preview

Služi za uvid u predloženi programski kod klase.

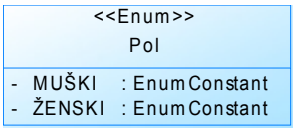


Generisani kod je proširen komentarima koji služe za Reverse engineering – proces kojim se tekuća verzija aplikacije usaglašava sa dijagramom klasa.

3.4 Enumeracije

Enumeracije se u UML-u predstavljaju kao klase, čiji atributi (tj. sam njihov naziv) služe za predstavljanje stavki enumeracije. Kreiraju se na sledeći način:

- kreirati klasu i dodeliti joj stereotip <<Enum>>
- stavke enumeracije dodati kao atribute, čiji tip je automatski podešen na EnumConstant.



Enumeracije mogu da imaju i obične atribute i metode.

3.5 Generičke klase

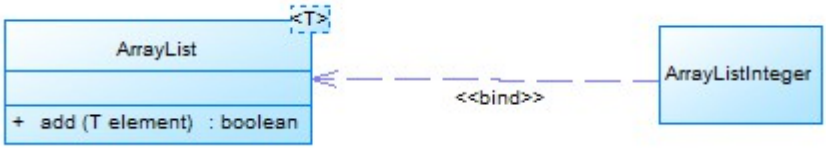
Generičke klase su klase koje poseduju parametre, generičke tipove, koji se koriste kao tip podataka za definisanje atributa i parametara metoda te klase, a koji se mogu zameniti konkretnim tipom podataka prilikom kreiranja drugih klasa ili instanci generičke klase. Na primer, sve kolekcije u Javi su generičke, a prilikom njihovog instanciranja se definiše tip podataka koji se u njih smešta:

ArrayList<Integer> ocene = new ArrayList<Integer>;

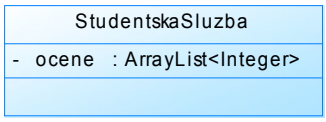
Generička klasa se tako što se u polju Type u dijalogu iznad odabere opcija Generic. Tada se pojalvjuje kartica Generic, na kojoj se definišu parametri klase. Ovi parametri će biti uključeni u poniđene tipove podataka prilikom unosa atributa i operacija klase.

Povezana (*bound*) klasa je klasa koja generičku klasu konkretizuje dodeljivanjem konkretnih tipova podataka generičkim parametrima. Kreira se na sledeći način:

- za Type odabrati Bound
- na kartici Generic svakom generičkom tipu dodeliti konkretan tip
- klasu povezati sa generičkom klasom vezom zavisnosti sa stereotipom <<bind>>.

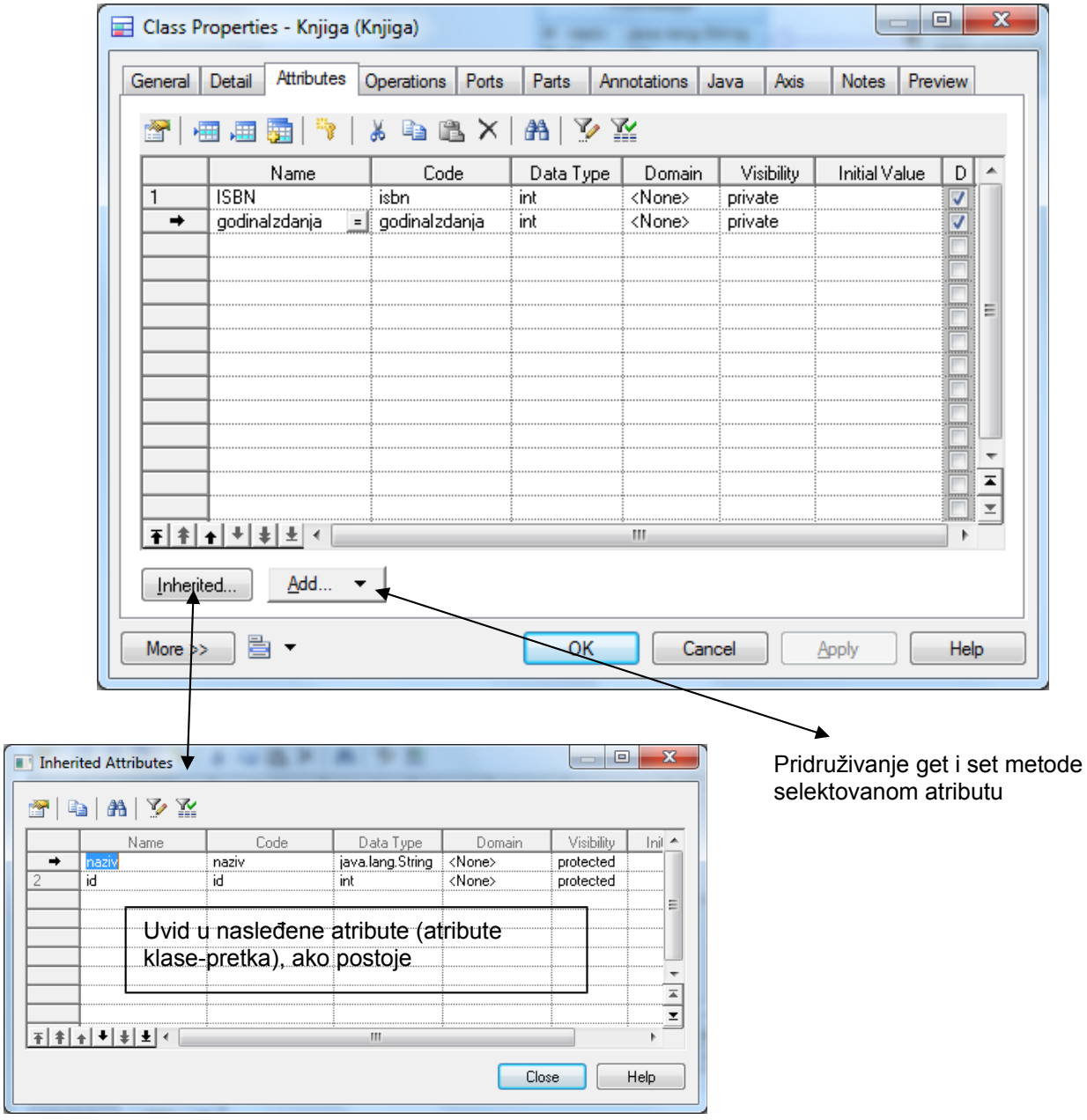


Drugi način korišćenja generičkih klasa podrazumeva definisanje konkretnih tipova za njene parametre prilikom instanciranja (kao u listi ocena na početku odeljka), tj. bez posrednika u obliku *bound* klase:



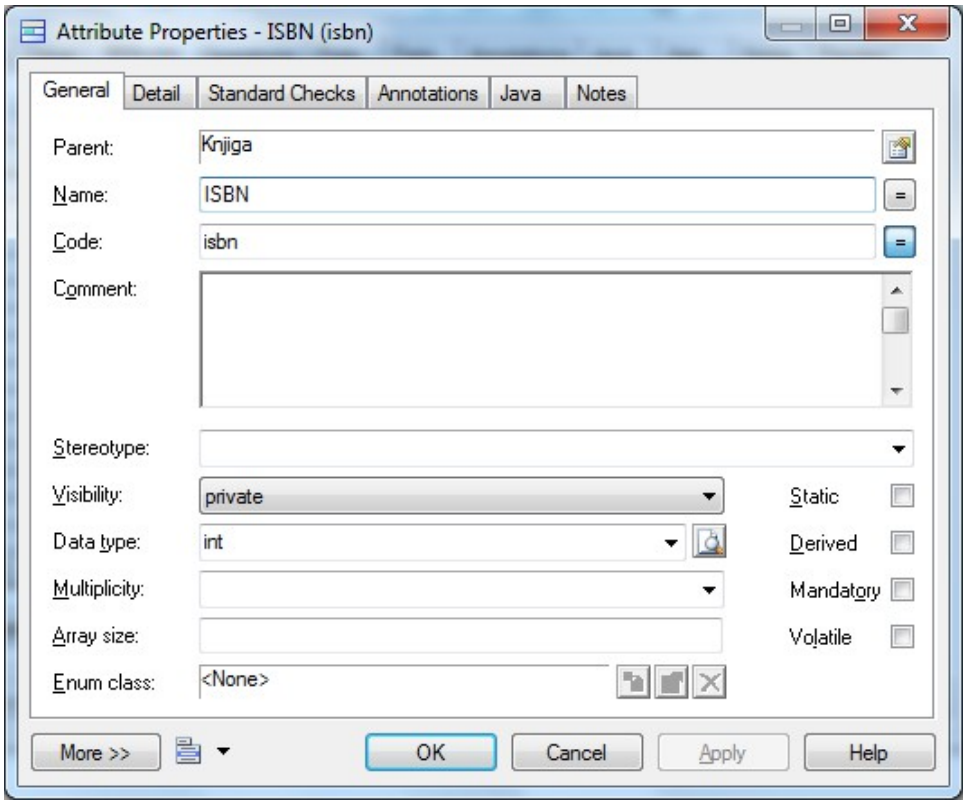
Ovakva klasa se ne povezuje sa generičkom klasom na dijagramu.

3.6 Atributi klase



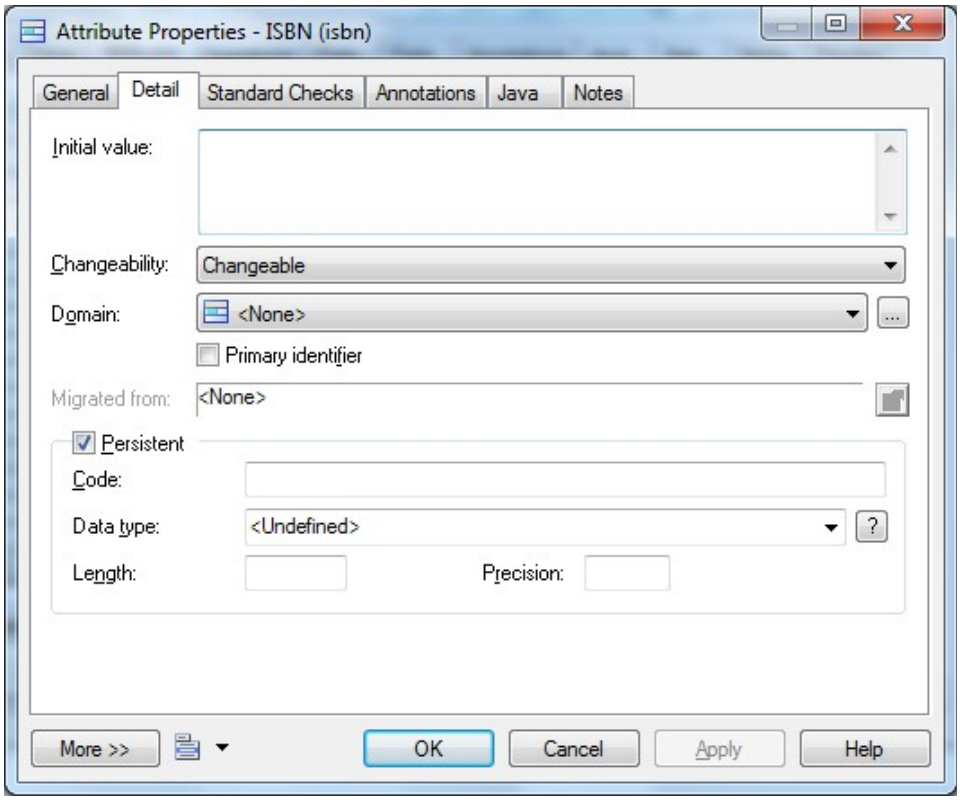
Kolona “D” u spisku atributa označava da li je atribut vidljiv u okviru dijagrama. Radi povećanja preglednosti dijagrama, moguće je ostaviti da budu vidljivi samo najvažniji atributi. *Vidljivost atributa se tiče samo izgleda dijagrama, ne i generisanja koda ili prava pristupa atributu!*

3.4.1 Kartica General



Name	Deskriptivan naziv atributa
Code	Naziv atributa koji se prenosi u programski kod
Visibility	Vidljivost atributa (public, protected, private, package). Podrazumevana vidljivost može se definisati u dijalogu Tools → Model Options.
Data Type	Tip atributa. Lista raspoloživih tipova zavisi od izabranog programskog jezika. Podrazumevani tip može se definisati u dijalogu Tools → Model Options.
Mulitplicity	Višestrukost atributa: opseg broja vrednosti koje objekat posmatrane klase može da ima za dati atribut (npr. 0..*)
Array size	Služi za specificiranje višestrukosti atributa kao slobodnog teksta. Ako ponuđene opcije u Multiplicity polju nisu dovoljne, ovde se može uneti opisni izraz ili izraz na ciljnom programskom jeziku (ne utiče na generisani kod).
Static	Oznaka da je atribut static (vezan za klasu, a ne za instancu klase)
Derived	Vrednost atributa može da se izvede iz vrednosti ostalih atributa
Mandatory	Obavezan atribut (donja granica za Multiplicity veća od nule)

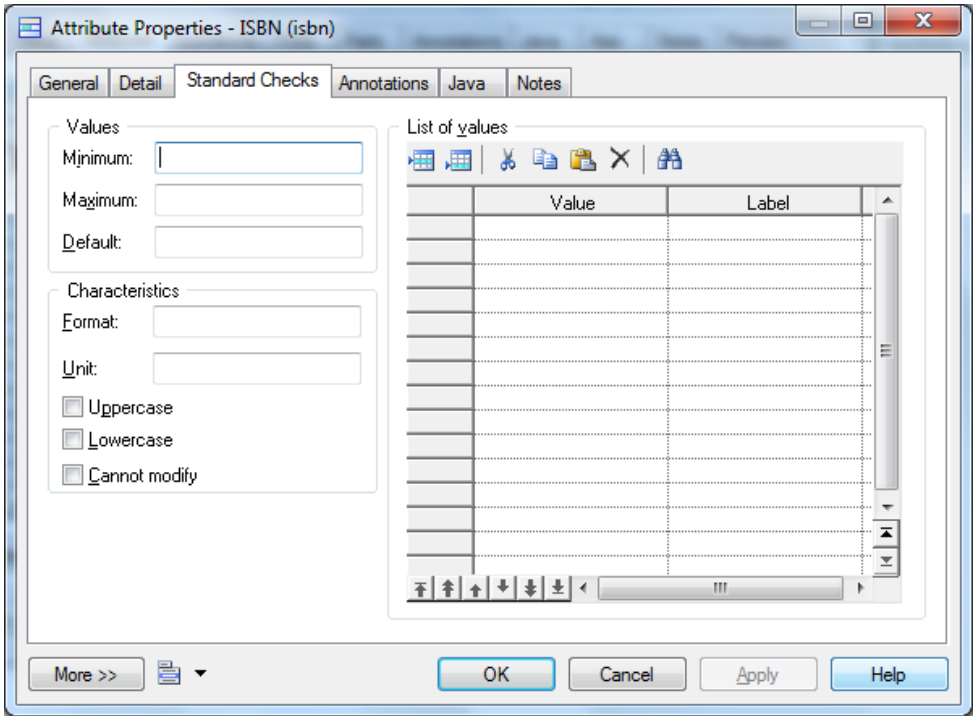
3.4.2 Kartica Detail



Initial value	Inicijalna vrednost atributa. Ukoliko je uneta, kôd za dodelu inicijalne vrednosti formira se u okviru konstruktora klase (C++), odnosno u odeljku za definisanje atributa (Java).
Changeability	Izmenljivost atributa nakon kreiranja instance klase: - Changeable : može da se menja (podrazumevano) - Read-only: može se samo očitavati (zabranjena set metoda) - Frozen: jednom dodeljena vrednost atributu ne može se menjati (konstanta) - Add-only (samo ako je višestrukost veća od 1): nove vrednosti se mogu dodavati, ali se ne mogu vršiti druge izmene kolekcije
Domain	Odabir domena, kojim se može zameniti tip atributa (v. odeljak 3.4.4 Domeni)
Primary identifier	Oznaka da se atribut pretvara u deo primarnog ključa u relacionom modelu
Persistent	Oznaka da je atribut perzistentan, tj. da predstavlja obeležje u tabeli koja odgovara posmatranoj klasi, sa sledećim osobinama: - Code: naziv obeležja - Data type: tip podatka obeležja

3.4.3 Kartica Standard Checks

Izmene na ovoj kartici ne utiču na generisani kod, već na generisanje konceptualnog/fizičkog modela.



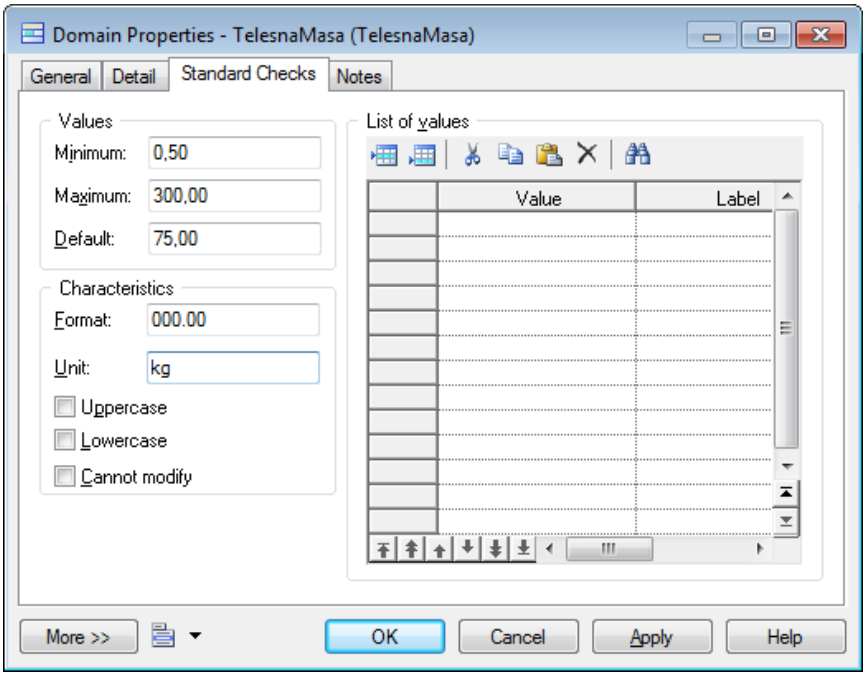
Values	Opseg dozvoljenih vrednosti (minimum i maksimum) i podrazumevana (default) vrednost atributa
Characteristics	Ostale osobine: - Format (npr. dd.mm.yy) - Unit: jedinica mere - Uppercase/Lowercase: dozvoljena samo velika/mala slova - Cannot modify: vrednost atributa se ne može menjati
List of values	Tabela u kojoj se mogu navesti dozvoljene vrednosti atributa. Value je dozvoljena vrednost, a Label opis.

3.4.4 Domeni

Domeni se u *PowerDesigner*-u koriste za definisanje templejta za tipove podataka. Domen predstavlja tip podataka sa pridruženim informacijama o ograničenjima i perzistentnosti.

Dodavanje domena:

1. iz Browser-a: desni klik na model → New → Domain
2. iz liste svih domena, koja može da se otvori iz glavnog menija (Model → Domains) ili sa Details kartice bilo kojeg atributa (v. odeljak 3.4.2 Kartica Detail).

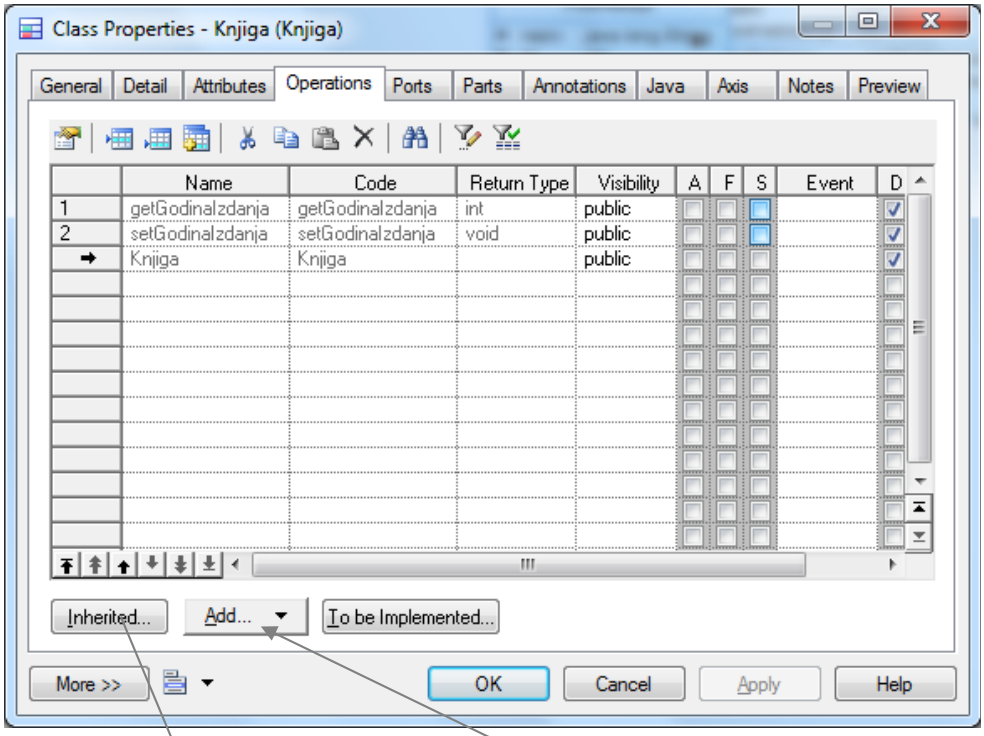


Osobine domena (koje se prenose na atribut kome se pridruži dati domen):

- tip podatka
- kardinalitet (multiplicity)
- sve osobine koje se definišu na kartici Standard Checks
- podaci o perzistentnosti atributa
- pridružena pravila (kartica Rules).

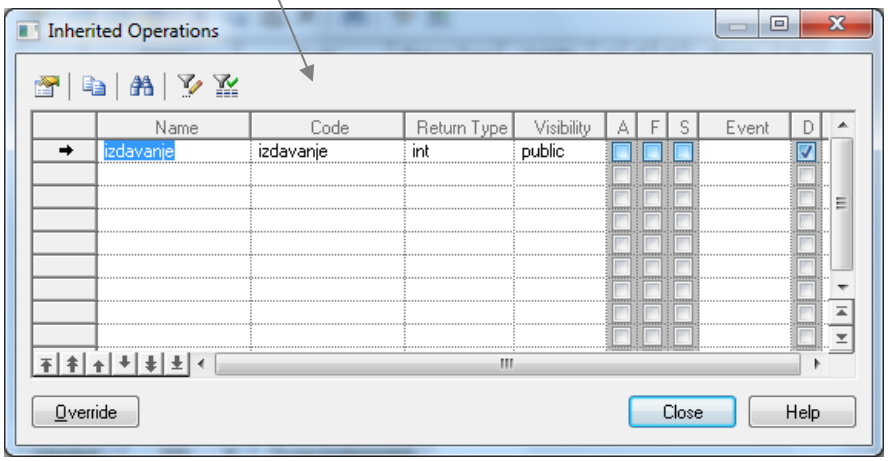
Kada se napravi izmena na domenu, ona se propagira na sve atribute tog domena.

3.5 Operacije klase



Kreiranje:

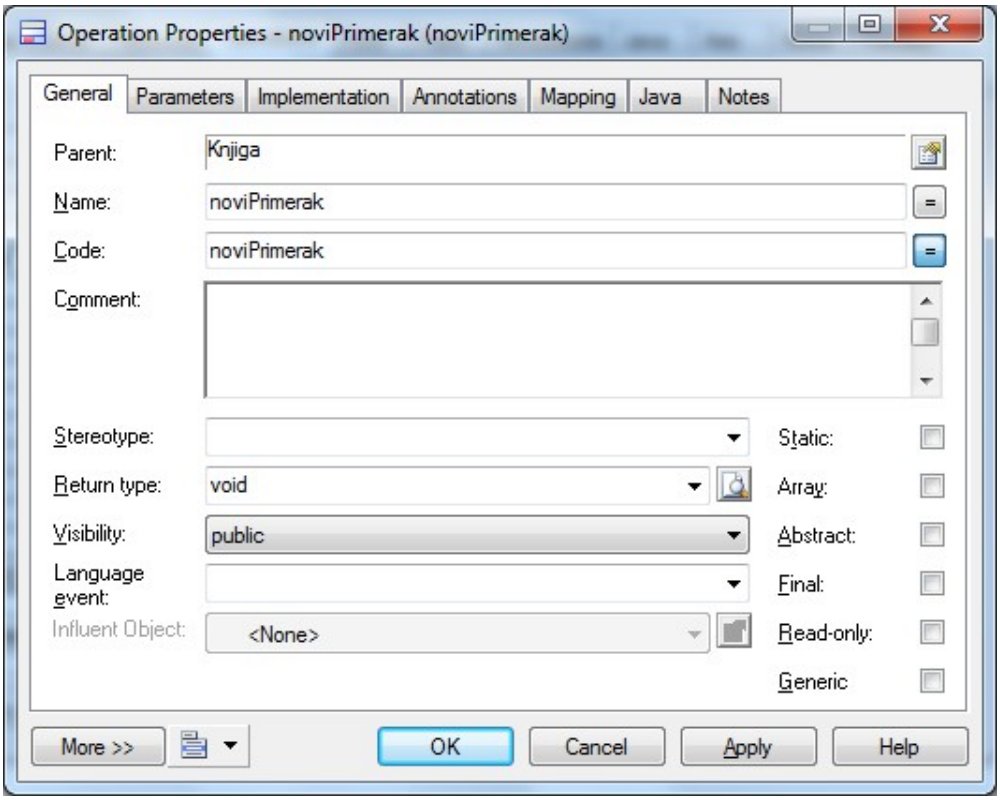
- podrazumevanog konstruktora/destruktora za klasu
- copy konstruktora
- bloka za inicijalizaciju
- operacije za kreiranje kopije objekta za koji je pozvana



Dijalog za uvid u operacije nasleđene od klase-pretka (ako postoji). Dugme Override omogućava preuzimanje izabranih operacija klase-pretka radi redefinisanja u okviru tekuće klase

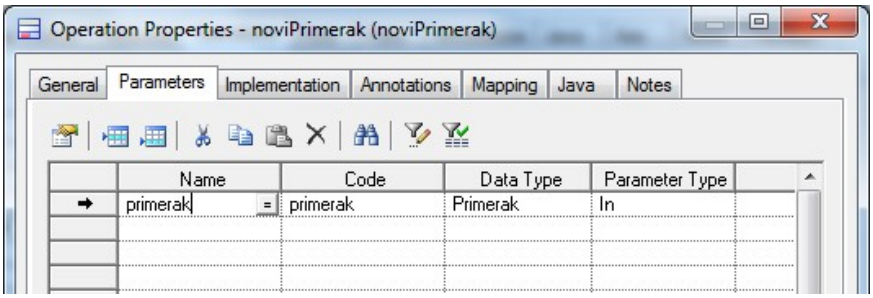
Dugme To be Implemented... omogućava uvid u operacije "nasleđene" od interfejsa (ako klasa realizuje neki interfejs), sa mogućnošću da se te operacije implementiraju. Dijalog je identičan dijalogu za uvid u operacije nasleđene od klase-pretka, samo se umesto dugmeta Override pojavljuje dugme Implement.

3.5.1 Kartica General

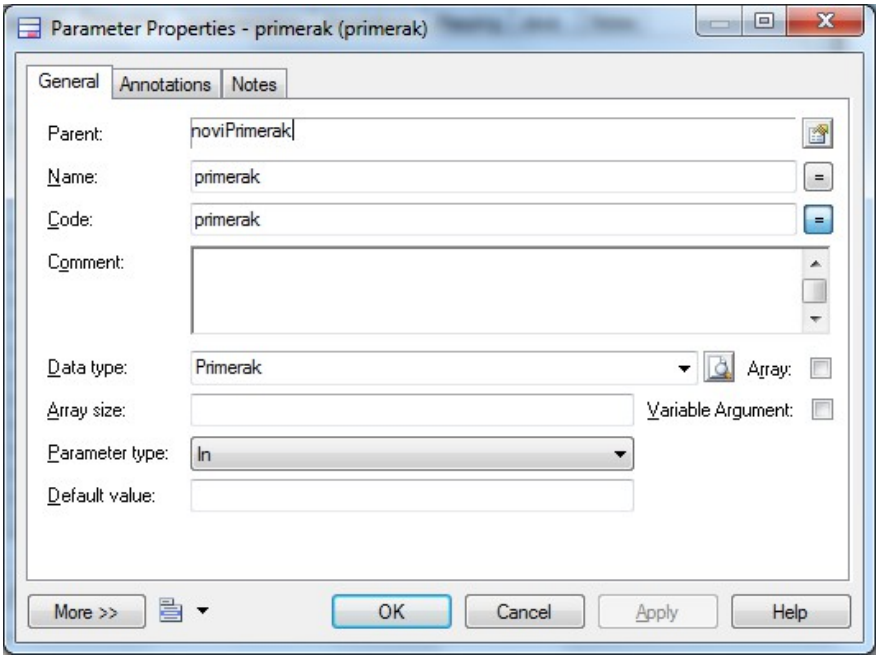


Name, Code	Naziv operacije (opisni i naziv koji se prenosi u generisani kod)
Return Type	Tip rezultata operacije. Ako operacija vraća niz, ovde se navodi tip elementa niza, a treba selektovati indikator Array. Podrazumevani tip može se definisati u dijalogu Tools → Model Options.
Visibility	Vidljivost operacije: public, protected, private i package
Language event	Naziv događaja koji aktivira datu operaciju (ukoliko se operacija definiše kao reakcija na neki događaj)
Influent Object	Ukoliko je operacija redefinisana ili implementira operaciju interfejsa, ovde je navedena originalna operacija klase-pretka odnosno interfejsa.
Static	Oznaka da je operacija statička
Array	Operacija vraća niz (tip elementa niza se definiše u polju Return Type)
Abstract	Oznaka da je operacija apstraktna (apstraktne operacije ne mogu da imaju implementaciju).
Final	Oznaka da nije dozvoljeno redefinisanje operacije u okviru klasa-naslednica (ako postoje)
Read-only	Operacija ne menja objekat za koji je pozvana
Generic	Operacija sa generičkim parametrima (definišu se na kartici Generic)

3.5.2 Kartica Parameters

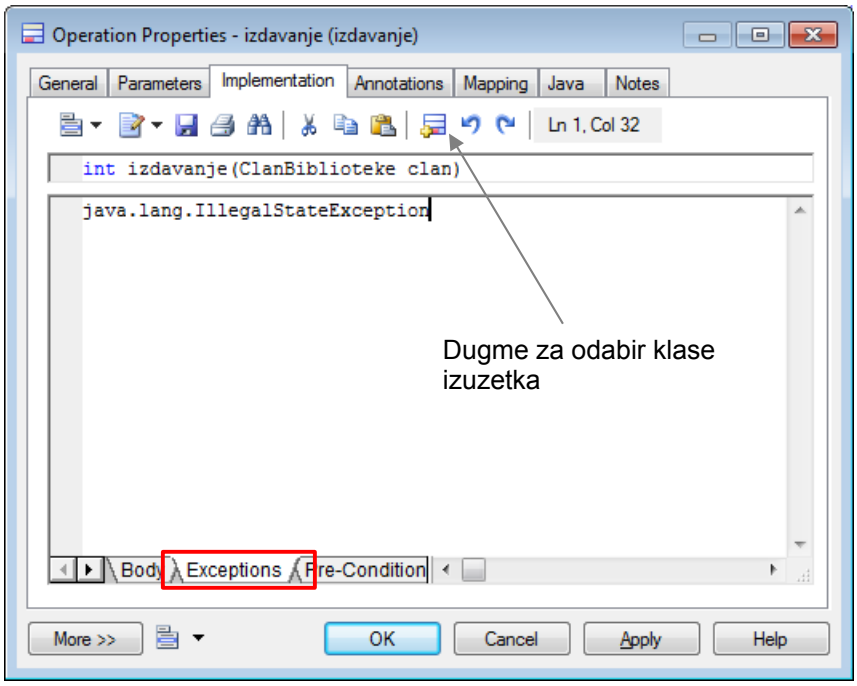


Dijalog za podešavanje osobina parametra:



Name, Code	Naziv parametra (opisni i naziv koji se prenosi u generisani kod)
Data Type	Tip parametra. Ako je u pitanju niz, treba selektovati indikator Array. U polju Array size može se navesti tačan broj elemenata niza.
Parameter type	Vrsta parametra: ulazni, izlazni, ulazno-izlazni
Default value	Podrazumevana vrednost parametra
Variable Argument	Selektovanjem ove opcije se definiše da operacija može da primi promenljiv broj argumenata. Opcija se selektuje samo za jedan parametar metode i on mora da bude poslednji u listi parametara.

3.5.3 Kartica Implementation



U donjem delu se nalazi niz kartica:

- body: definisanje tela operacije
- specification, pre-condition, post-condition: tekstualni opis ili pseudo-kod koji opisuje operaciju, njene preduslove i uslove koji važe nakon njenog završetka
- exceptions: Na ovoj kartici se mogu definisati izuzeci koje operacija baca (v. sliku iznad). Klasa izuzetka se može uneti ručno ili se može izabrati neka od klasa u modelu. Ovako definisani izuzeci se dodaju deklaraciji operacije (u okviru *throws* klauzule) prilikom generisanja koda.

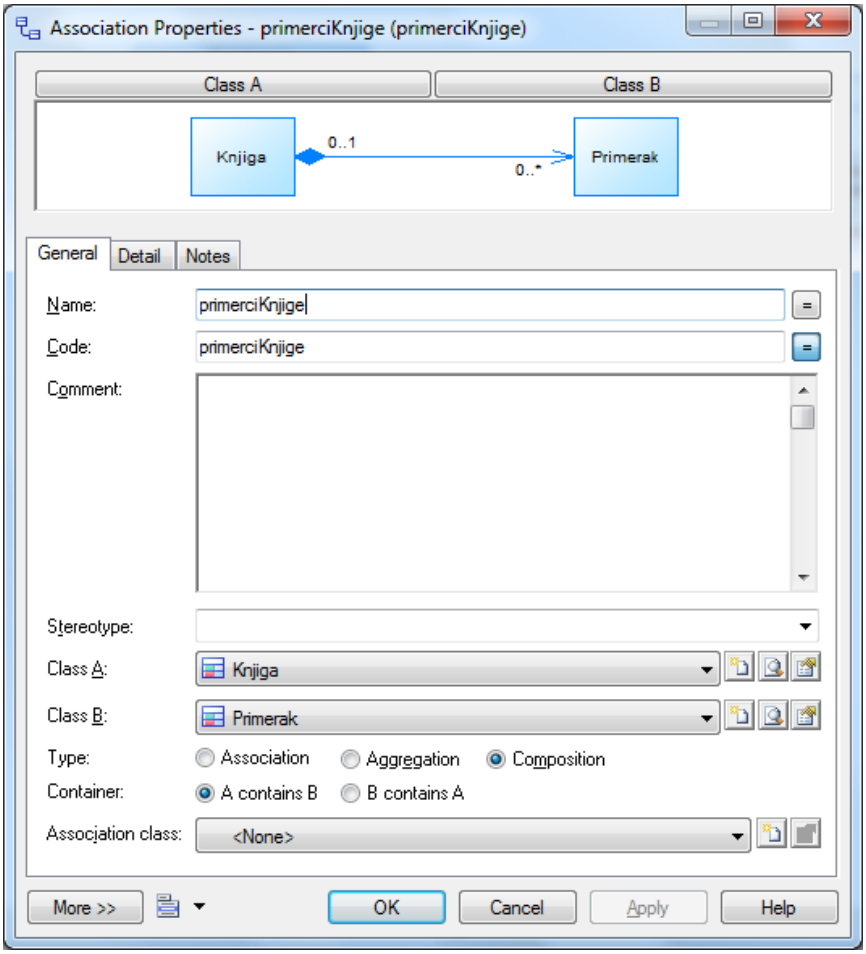
4. Dijalozi za podešavanje osobina veza

4.1 Asocijacija

Asocijacija je strukturna veza koja definiše odnos udruživanja i/ili posedovanja između instanci klasa. Preko asocijacije klase A sa klasom B instance klase A mogu doći do instanci klase B, što se obično implementira uvođenjem atributa (reference na B) u klasi A. Asocijacija može da bude i dvosmerna.

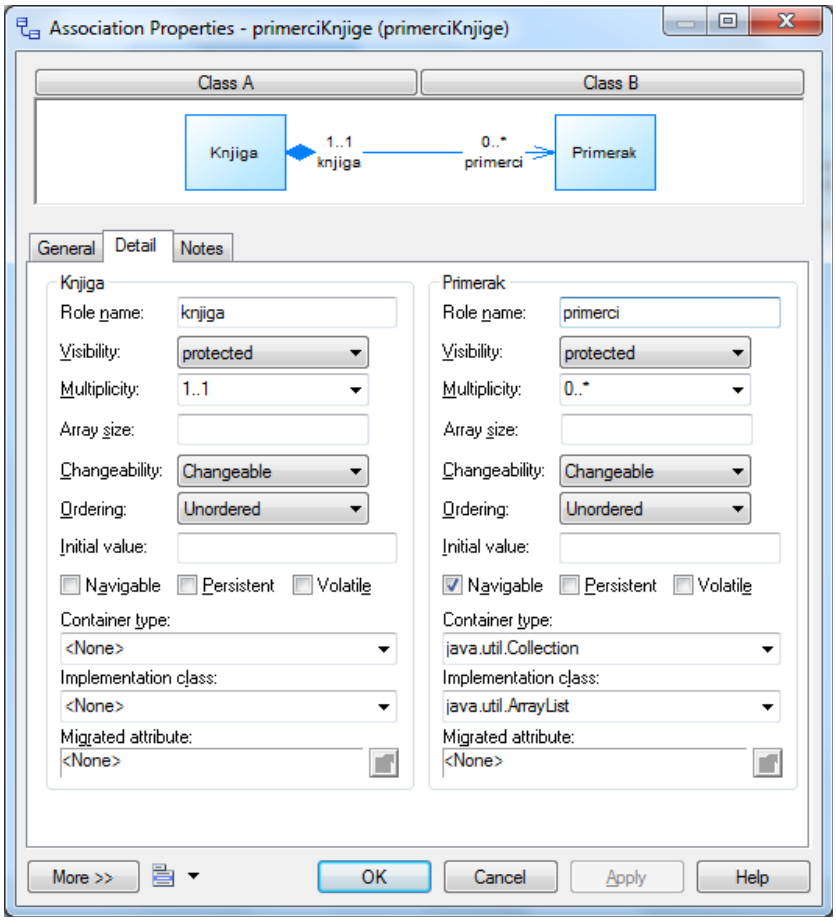
- Obična asocijacija je uopštena strukturna veza, koja se koristi za modelovanje udruživanja bilo koje dve klase.
- Agregacija je specijalna vrsta asocijacije koja modeluje odnos "celina-delovi", pri čemu su delovi samostalne klase čije instance mogu da "žive" i nezavisno od klase koja ih agregira.
- Kompozicija je specijalna vrsta agregacije kod koje su delovi slabi objekti koji ne mogu da samostalno postoje bez osnovne klase.

4.1.1 Kartica General



Name, Code	Deskriptivan i kodni naziv veze. Ne utiču na generisani kod.
Type	Tip asocijacije (obična, agregacija ili kompozicija). Tip je unapred definisan odabirom jedne od tri alatki sa palete, a ovde se može naknadno promeniti.
Association class	Klasa vezana za datu asocijaciju (v. odeljak 4.1.3 Klasa asocijacije)

4.1.2 Kartica Detail



Za svaku stranu (ulogu) asocijacije definišu se sledeći parametri:

Role 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.
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
Multiplicity	Kardinalitet veze (donja i gornja granica)
Array size	Ako je Multiplicity veći od 1, ovde se može navesti tačan broj elemenata niza
Changeability	Izmenljivost odgovarajućeg atributa u okviru suprotne klase. Ima iste moguće vrednosti i značenja kao izmenljivost običnih atributa (v. odeljak 3.4.2 Kartica Detail).
Ordering	*Samo ako je gornji kardinalitet veći od 1*: Definiše da li su instance sa posmatrane strane veze uređene (*sorted* – sortirane prema nekoj vrednosti ili *ordered* – uređene po nekom drugom, spolja definisanom kriterijumu) ili neuređene (*unordered*).
Initial value	Početna vrednost atributa. Može da bude i izraz za inicijalizaciju, npr. `new Knjiga()` (prenosi se u generisani kod).
Navigable	Oznaka da li instanca jedne klase može da pristupi instanci druge klase
Persistent	Oznaka da li je asocijacija (tj. odgovarajući atribut) perzistentan
Container type, Implementation class	Odabir kontejnera koji se koristi za smeštanje vrednosti višestrukog atributa. Ponuđene vrednosti zavise od ciljnog jezika, a moguće je i uneti proizvoljan naziv. Podrazumevani kontejner može se definisati u dijalogu Tools → Model Options, polje Default association container.

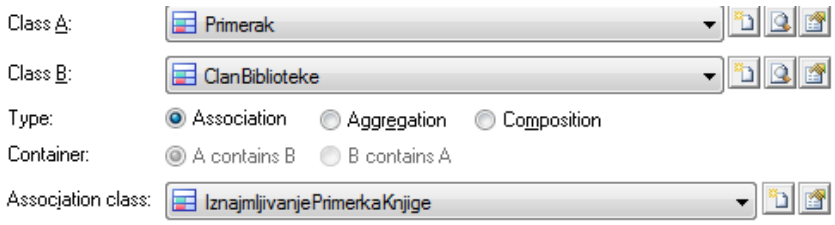
4.1.3 Klasa asocijacije

Klasa asocijacije predstavlja asocijaciju koja poseduje karakteristike klase, odnosne ima atribute i metode, a moguće su i veze sa drugim klasama. Klasa asocijacije označava se kao "obična" klasa, ali je povezana sa odgovarajućom asocijacijom isprekidanom linijom.

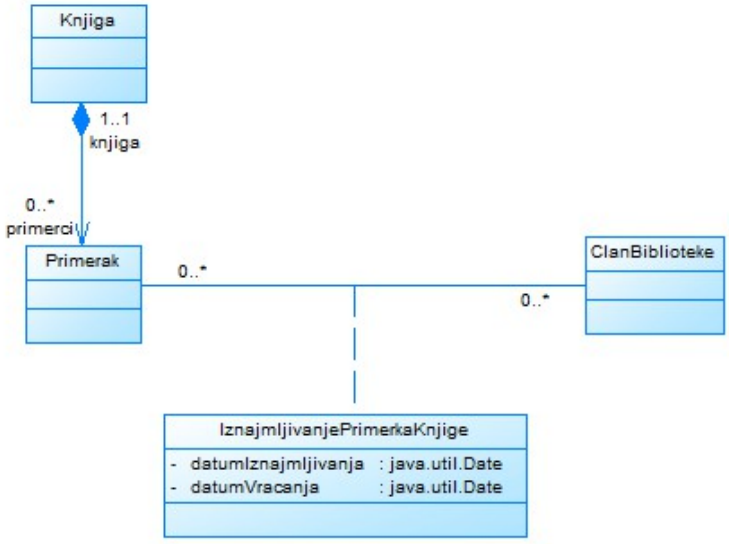
U *PowerDesigner*-u, klasa asocijacije se kreira u dva koraka:

1. Kreira se klasa, sa svojim atributima i metodama
2. U Properties dijalogu za asocijaciju, na kartici General, u polju Association class odabere se klasa koja treba da je vezana za asocijaciju.

Primer: Član biblioteke je povezan sa primerkom knjige koju želi da iznajmi vezom asocijacije, koja je dopunjena klasom asocijacije u kojoj se nalaze podaci o datumu iznajmljivanja i datumu vraćanja primerka.



Dijalog asocijacije, kartica General, polje Association class

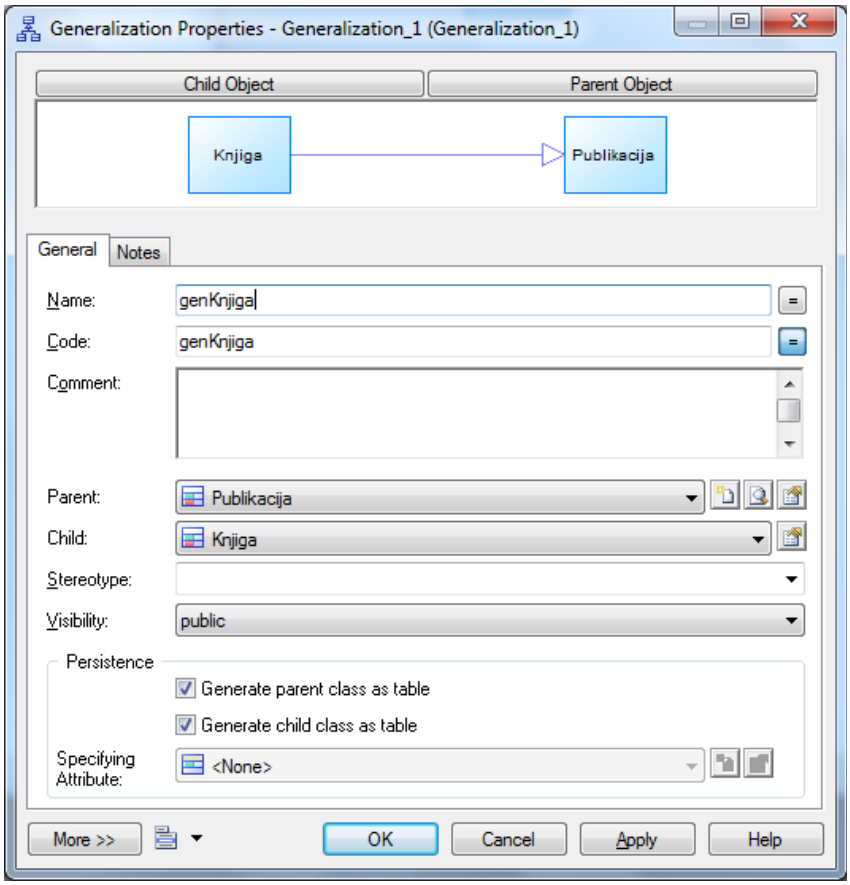


Odgovarajući dijagram klasa

4.2 Generalizacija

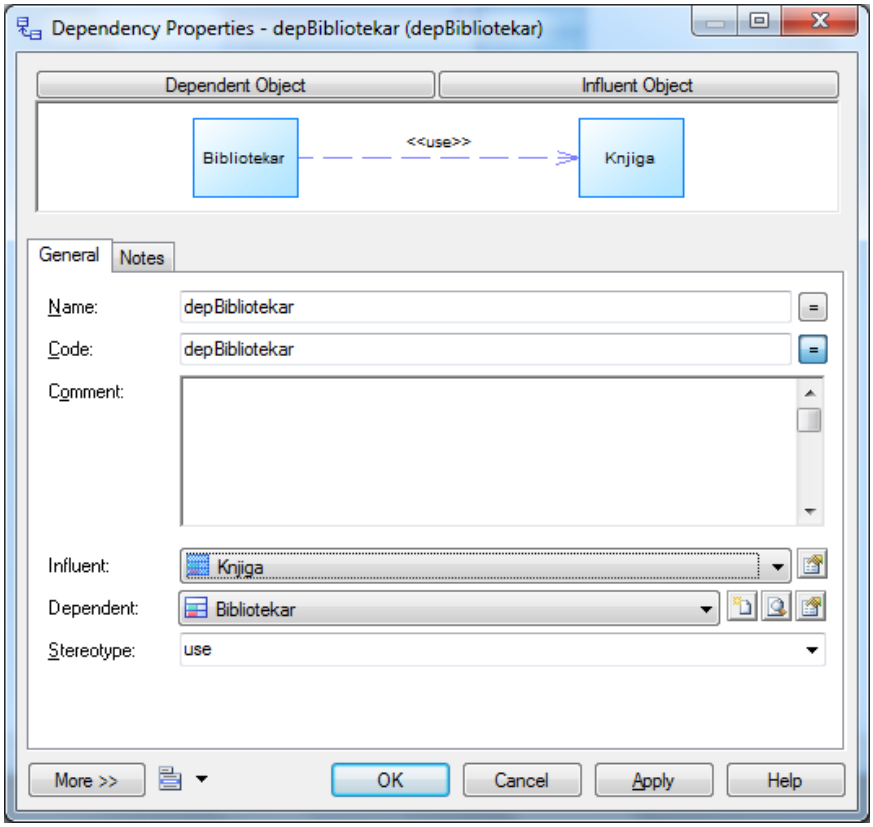
Služi za modelovanje hijerarhijske veze (nasleđivanja) između klasa. U dijalogu je moguće podesiti:

- Visibility: vidljivost nasleđivanja
- Persistence: način na koji će veza biti preslikana u konceptualni/fizički model.



4.3 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.

Specijalan slučaj veze zavisnosti može biti predstavljen navođenjem odgovarajućeg sterotipa. Postoji niz stereotipa koji su definisani za vezu zavisnosti, a korisnik može i da definiše svoje.
Neki predefinisani stereotipovi su:

- *use* – funkcionalnost ili implementacija jednog elementa modela zahteva prisustvo drugog elementa modela (npr. jedna klasa koristi objekat druge klase kao parametar svoje metode). Služi za naglašavanje osnovnog tipa veze zavisnosti u odnosu na sve ostale stereotipove.
- *bind* – veza parametrizovane (template) klase i klase koja daje konkretne vrednosti parametrima (v. odeljak 3.5 Generičke klase)
- *friend* – modelovanje "prijateljskih" klasa
- *instantiate* – operacije nezavisne klase kreiraju instance zavisne klase.

4.4 Realizacija

Na dijagramu klasa se koristi za označavanje veze između interfejsa i klase koja ga implementira.