IMPLEMENTAIJA ODNOSA SA DIJAGRAMA KLASA U JAVA KODU

Uvod

Kod tradicionalnog modela podatika (dijagram "entiteti i odnosi" ili "model objekata i odnosa") objekti kao instance klasa su mogli da stupaju u tri vrste odnosa:

- učešće u vezi (bez ili sa svojstvima);
- učešće u specijalizaciji;
- učešće u zavisnosti.

Kod objektnog modela podatiKa (dijagram klasa) navedene tri vrste osnosa se redom predstavljaju sa UML odnosima:

- asocijacija;
- specijalizacija;
- agregacija.

U daljem razmatranju smatraće se da su svi posmatrani odnosi između klasa bidirekcioni, odnosno da postoji navigacija u oba smera, a na kraju će biti data napomenama o implikacijama usmerenosti odnosa. Koristi se sledeća notacija:

Klasa	neka klasa Klasa;
rKlasa	referenca na neku klasu Klasa;
iKlasa	instanca neke klase Klasa prilikom kreiranja ili kao argument.

Osnovni mehanizam implementacije odnosa između klasa

Učešće neke klase A u nekom odnosu sa drugom klasom B implementira se preko dinamičkog svojstva reference na drugu klasu B u odnosu, i obrnuto. U slučaju da je gornja kardinalnost učešća klase A u odnosu jedniKa 1, referentno svojstvo je skalarna referenca na klasu B, a u slučaju da je jedniKa N, referentno svojstvo je kolekcija referenci na klasu B. Ovo važi za sve odnose koji nemaju svojstva. Specijalni slučaj asocijacije sa svojstvima, odnosno asocijativne klase, biće posebno obrađen.

S obzirom da instanca neke klase A ne može više puta biti u nekom odnosu sa istom instancom neke klase B, i obrnuto, kao prirodni izbor za kolekciju referenci nameće se skup (**Set<*E*>**) čija deklaracija sa bitnim metodima glasi:

```
public interface Set<Klasa> extends Collection<Klasa>
   {
   int size();                        // broj elemenata
   boolean isEmpty();                 // da li je prazan
   boolean contains(Klasa element);   // da li sadrzi
   boolean add(Klasa element);        // dodavanje datog elementa
   boolean remove (Klasa element);    // uklanjanje zadatog elementa
   ...
   }
```

Od tri postojeće implementacije ovog interfejsa u Java platformi, po performansama je optimalna **HashSet**. Kreiranje skupa referenci na neku klasu **KlasaX** se postiže sa:

```
Set<KLasaX> Skup = new HashSet<KlasaX> ();
```

Nakon toga, neka instanca i**KlasaX** se dodaje u skup sa

```
bollean Uspeh = Skup.add ( iKlasaX );
```

a uklanja iz njega sa

```
boolean Uspeh = Skup.remove ( iKlasaX );
```

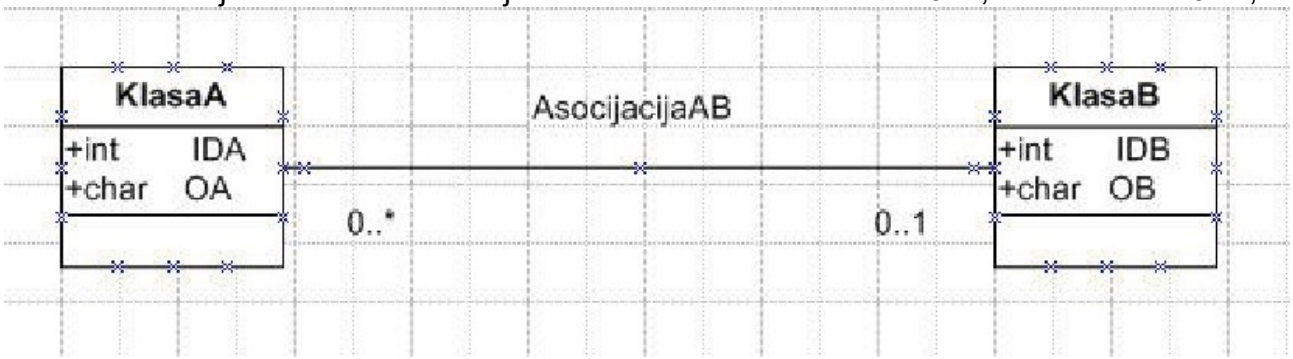
Implementacija pojedinih vrsta odnosa

Za svaki konkretni slučaj dat je dijagram klasa i nakon toga odgovarajući generisani Java kod, pri čemu su crveno označeni delovi koji proizilaze iz odnosa između klasa.

Asocijacija

Asocijacija bez svojstava sa kardinalnostima učešća 0..1 - 0..N

Za slučaj da klasa A učestvuje u odnosu sa kardinalnošću 0..1, a klasa B sa 0..N,



generisani Java kod će glasiti:

```
import ... KlasaB;
public class KlasaA
  {
  int    IDA;
  char   OA;
  KlasaB rAsocijacijaAB;
  }

import ... KlasaA;
public class KlasaB
  {
  int          IDB;
  char         OB;
  Set<KlasaA> rAsocijacijaAB = new HashSet<KlasaA> ();
  }
```

Za dva učesnika, iKlasaA i iKlasaB, odnos asocijacije se uspostavlja sa

```
iKlasaA.rAsocijacijaAB = iKlasaB;
boolean Uspeh = iKlasaB.rAsocijacijaAB.add ( iKlasaA );
```

a raskida sa

```
iKlasaA.rAsocijacijaAB = null;
boolean Uspeh = iKlasaB.rAsocijacijaAB.remove ( iKlasaA );
```

Ovo se može implementira kao dinamički metodi bilo koje od klasa učesnica u odnosu. U slučaju da je to KlasaA i da su u pitanju instance iKlasaA i iKlasaB, odgovarajući metodi u KlasaA će glasiti

```
void UspostaviAsocijaciju ( KlasaB iKlasaB )
  {
  rAsocijacijaAB = iKlasaB;
  boolean NeTreba = iKlasaB.rAsocijacijaAB.add ( this );
  return;
  }

void RaskiniAsocijaciju ( KlasaB iKlasaB )
  {
  rAsocijacijaAB = null;
  boolean NeTreba = iKlasaB.rAsocijacijaAB.remove (this);
  return;
  }
```

što se u klasi kontroloru slučaja upotrebe (i svugde gde eventualno treba) u okviru logike slučaja upotrebe unutar konstruktora kontrolora poziva sa

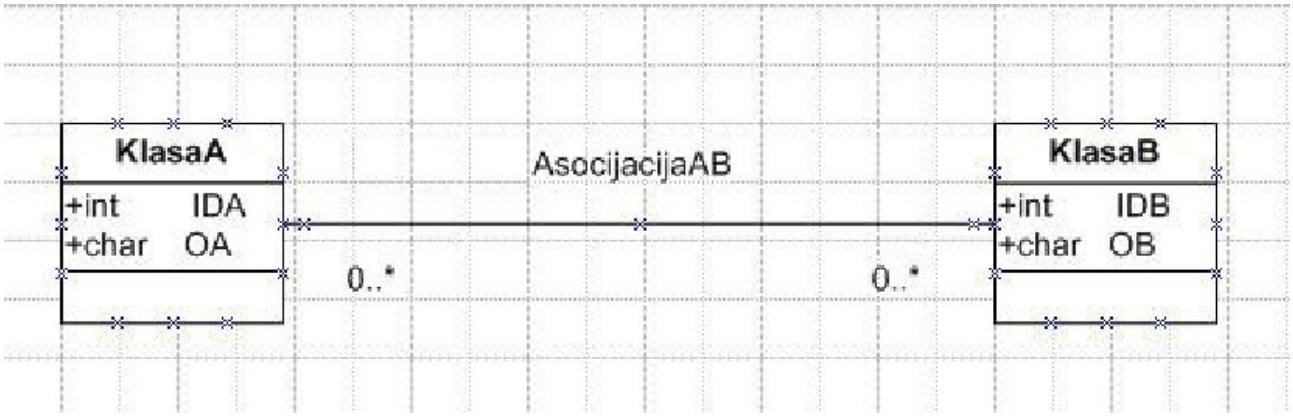
```
iKlasaA.UspostaviAsocijaciju ( iKlasaB );
```

odnosno za rskidanje sa

```
iKlasa.RaskiniAsocijaciju ( iKlasaB );
```

Varijanta sa metodima u KlasaB je simetrična.

Asocijacija bez svojstava sa kardinalnostima učešća 0..1 - 0..N



Za ovaj slučaj generisani Java kod će glasiti

```
import ... KlasaB;
public class KlasaA
  {
  int    IDA;
  char   OA;
  Set<KlasaB> rAsocijacijaAB = new HashSet<KlasaB> ();
  }

import ... KlasaA;
public class KlasaB
  {
  int         IDB;
  char        OB;
  Set<KlasaA> rAsocijacijaAB = new HashSet<KlasaA> ();
  }
```

Odnos asocijacije se uspostavlja sa

```
boolean Uspeh = iKlasaA.rAsocijacijaAB.add ( iKlasaB );
boolean Uspeh = iKlasaB.rAsocijacijaAB.add ( iKlasaA );
```

a raskida sa

```
boolean Uspeh = iKlasaB.rAsocijacijaAB.remove ( iKlasaA );
boolean Uspeh = iKlasaB.rAsocijacijaAB.remove ( iKlasaA );
```

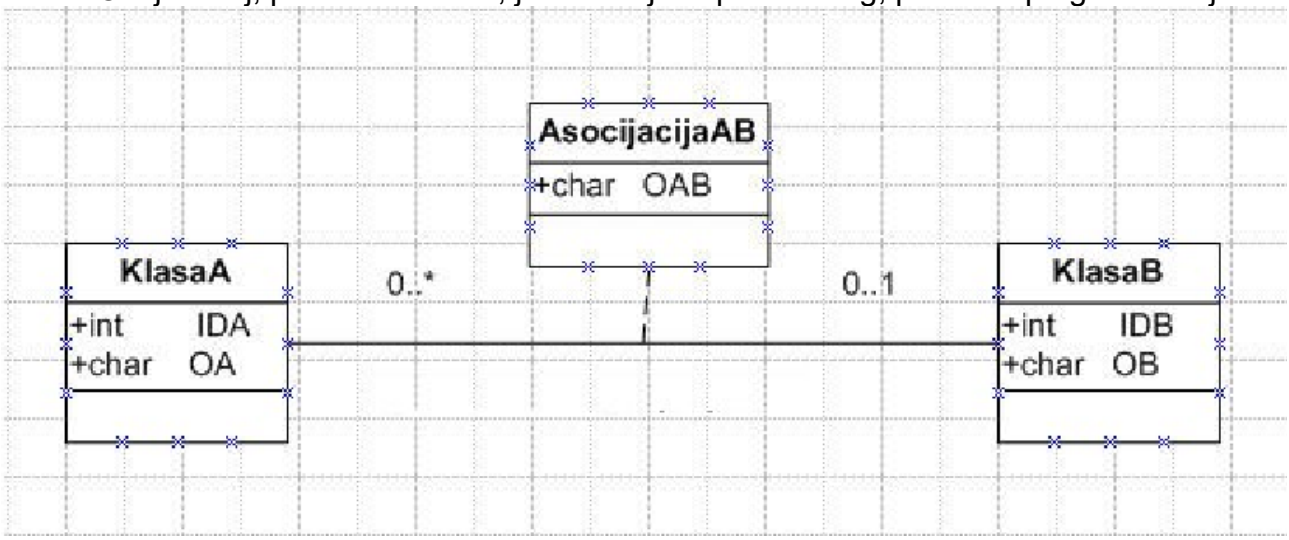
Odgovarajući dinamički metodi u KlasaA (varijanta u KlasaB je simetrična) je

```
void UspostaviAsocijaciju ( KlasaB iKlasaB )
  {
  boolean NeTreba = iKlasaA.rAsocijacijaAB.add ( iKlasaB );
  boolean NeTreba = iKlasaB.rAsocijacijaAB.add ( this );
  return;
  }
void RaskiniAsocijaciju ( KlasaB iKlasaB )
  {
  boolean NeTreba = iKlasaA.rAsocijacijaAB.remove ( iKlasaB );
  boolean NeTreba = iKlasaB.rAsocijacijaAB.remove ( this );
  return;
  }
```

pri čemu pozivi iz klase kontrolora slučaja (i eventualno još negde) ostaju kao i prethodni.

Asocijacija sa svojstvima sa kardinalnostima učešća 0..1 - 0..N

Ovaj slučaj, prikazan na slici, je složeniji od prethodnog, pošto će pri generisanju



Java koda pored klasa učesnica biti prisutna i asocijacijativna klasa. Isti rezultat bi dao i ekvivalentni dijagram klasa na kome su klase učesnici u odnosu posredno, preko asocijativne klase, pri čemu je svaka instanca te klase u odnosu sa po jednom instancom svake klase učesnika u odnosu. Generisani Java kod će glasiti:

```
import ... KlasaB
import ... AsocijacijaAB;
public class KlasaA
  {
  int           IDA;
  char          OA;
  AsocijacijaAB rAsocijacijaAB;
  }
```

```
import ... KlasaA;
import ... KlasaB;
public class AsocijacijaAB
  {
  char   OAB;
  KlasaA rKlasaA;
  KlasaB rKlasaB;
  }
```

```
import ... KlasaA;
import ... AsocijacijaAB;
public class KlasaB
  {
  int          IDB;
  char         OB;
  Set<AsocijacijaAB> rAsocijacijaAB = new HashSet<AsocijacijaAB>();
  }
```

Za dva učesnika (instance) iKlasaA i iKlasaB i zadato svojstvo vOAB, odnos asocijacije se uspostavlja sa

```
AsocijacijaAB iAsocijatijaAB = new AsocijacijaAB ();
aAsocijacijaAB.rKlasaA = iKlasaA;
aAsocijacijaAB.rKlasaB = iKlasaB;
aAsocijacijaAB.OAB = vOAB;
```

```
iKlasaA.rfAsocijacijaAB = iAsocijacijaAB;
```

```
boolean Uspeh = iKlasaB.rAsocijacijaAB.add ( iAsocijacijaAB );
```

a raskida sa

```
iKlasaA.rAsocijacijaAB = null;
```

```
boolean Uspeh = iKlasaB.rAsocijacijaAB.remove ( iAsocijacijaAB );
```

pri čemu će "garbage collector" prepoznati situaciju da na **aAsocijacijaAB** više nema ni jedne reference i sprovesti implicitni "dispose".

Uspostavljanje asocijacije se može implementirati u okviru konstruktora asocijativnog entiteta kao

```
AsocijacijaAB AsocijacijaAB
              ( KlasaA iKlasaA, KlasaB iKlasaB, char vOAB )
  {
  AsocijacijaAB iAsocijacijaAB = new AsocijacijaAB ();
  OAB = vOAB
  rKlasaA = iKlasaA;
  rKlasaB = iKlasaB;
  iKlasaA.rAsocijacijaAB = iAsocijacijaAB;
  iKlasaB.rAsocijacijaAB.add ( iAsocijacijaAB );
  return iAsocijacijaAB;
  }
```

što se u klasi kontroloru slučaja upotrebe (i svugde gde eventualno treba) u okviru logike slučaja upotrebe unutar konstruktora kontrolora poziva sa

```
AsocijacijaAB iAsocijacijaAB = new AsocijacijaAB ( iKlasaA, iKlasaB, vOAB );
```

Prethodna varijanta nije poželjna, pošto se po njoj kontrolor klasa obraća asocijativnoj klasi, što nije dobro rešenje (obraćanje treba da bude samo ka domenskim klasama, a one trebaju da budu odgovorne za detalje uspostavljanja odnosa). Zato je bolja varijanta kod koje se uspostavljanje asocijacije vrši preko metoda bilo koje od klasa učesnica.

U slučaju da je to KlasaA (varijanta za za KlasaB je simetrična), taj metod glasi

```
void UspostaviAsocijaciju
     ( KlasaB iKlasaB, char vOAB )
  {
  AsocijacijaAB iAsocijacijaAB = new AsocijacijaAB ();
  iAsocijacijaAB.OAB = vOAB;
  iAsocijacijaAB.rKlasaA = this;
  iAsocijacijaAB.rKlasaB = iKlasaB;
  rAsocijacijaAB = iAsocijacijaAB;
  iKlasaB.rAsocijacijaAB.add ( iAsocijacijaAB );
  return;
  }
```

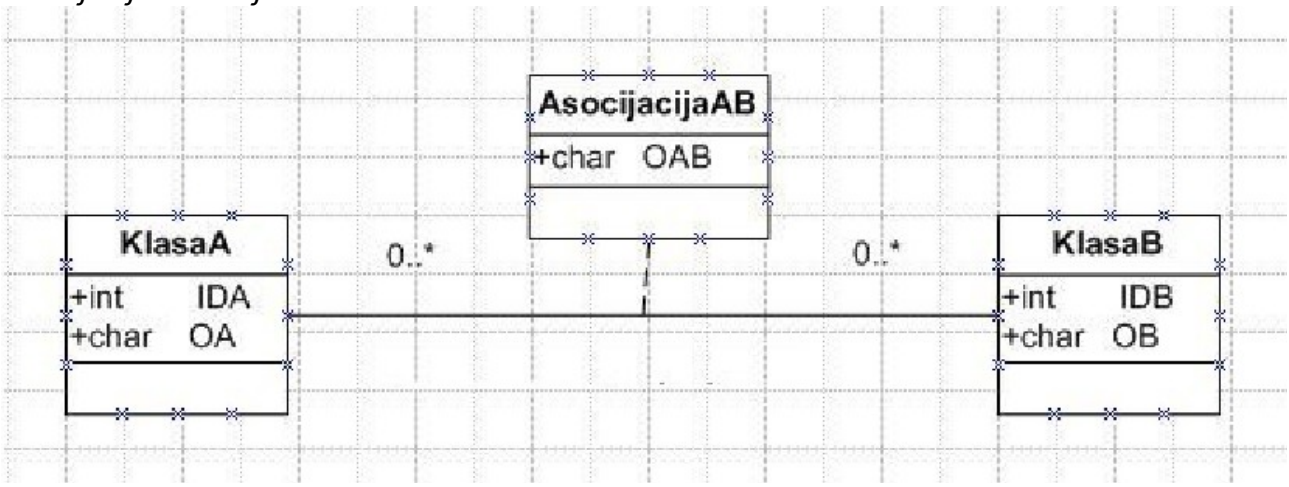
U slučaju prethodne poželjnije varijante uspostavljanja asocijacije, raskidanje asocijacije se implementira kao metod bilo koje od klasa učesnica. U slučaju da je to KlasaA (varijanta za za KlasaB je simetrična), taj metod glasi

```
void RaskiniAsocijaciju ( KlasaB iKlasaB )
  {
  AsocijacijaAB iAsocijacijaAB = rAsocijacijaAB;
  rAsocijacijaAB = null;
  iKlasaB.rAsocijacijaAB.remove ( iAsocijacijaAB );
  return;
  }
```

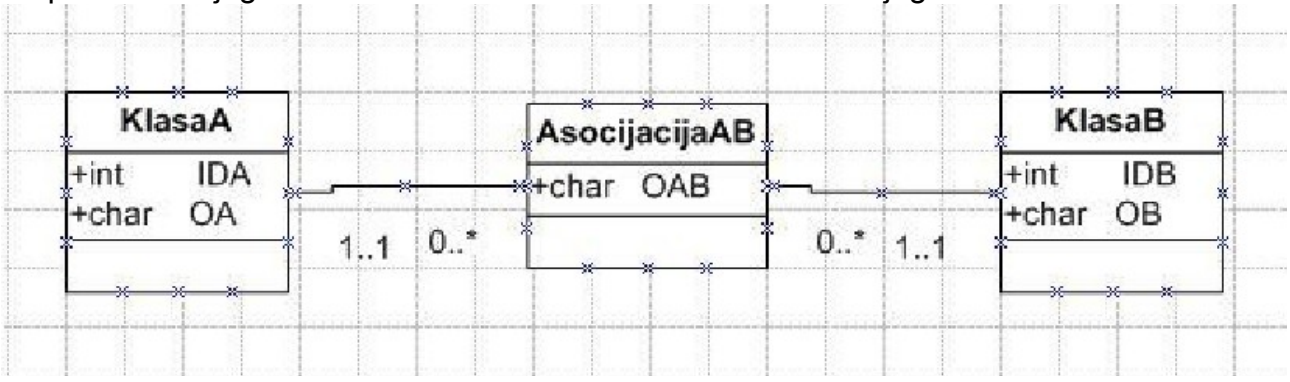
što se u klasi kontroloru slučaja upotrebe (i svugde gde eventualno treba) u okviru logike slučaja upotrebe poziva sa

```
iKlasaA.RaskiniAsocijaciju ( iKlasaB );
```

Asocijacija sa svojstvima sa kardinalnostima učešća 0..1 - 0..N



Za prethodni dijagram može se formirati sledeći ekvivalentni dijagram:



Generisani Java kod klasa je

```
import ... KlasaB
import ... AsocijacijaAB;
public class KlasaA
  {
  int           IDA;
  char          OA;
  Set<AsocijacijaAB> rAsocijacijaAB = new HashSet<AsocijacijaAB>();
  }
```

```
import ... KlasaA;
import ... KlasaB;
public class AsocijacijaAB
  {
  char   OAB;
  KlasaA rKlasaA;
  KlasaB rKlasaB;
  }
```

```
import ... KlasaA;
import ... AsocijacijaAB;
public class KlasaB
  {
  int         IDB;
  char        OB;
  Set<AsocijacijaAB> rAsocijacijaAB = new HashSet<AsocijacijaAB>();
  }
```

Za dva učesnika (instance) iKlasaA i iKlasaB i zadato svojstvo vOAB, odnos asocijacije se uspostavlja sa

```
AsocijacijaAB iAsocijatijaAB = new AsocijacijaAB ();
aAsocijacijaAB.rKlasaA = iKlasaA;
aAsocijacijaAB.rKlasaB = iKlasaB;
aAsocijacijaAB.OAB = vOAB;
boolean Uspeh = iKlasaA.rAsocijacijaAB.add ( iAsocijacijaAB );
boolean Uspeh = iKlasaB.rAsocijacijaAB.add ( iAsocijacijaAB );
```

a raskida sa

```
boolean Uspeh = iKlasaA.rAsocijacijaAB.remove ( iAsocijacija
boolean Uspeh = iKlasaB.rAsocijacijaAB.remove ( iAsocijacijaAB );
```