**TEHNIKA I INFORMATIKA U OBRAZOVANJU**

**Konferencija 32000 Čačak 9-11. Maja 2008.**

UDK: 004 : 371

Stručni rad

# VEZA ZAVISNOSTI REALIZACIJA

*Munir Šabanović [1], Momčilo Vujičić [2]*

***Rezime**: Veza realizacija spada u veze zavisnosti koje opisuju vezu između snabdevača i klijenta, gde ponašanje klijenta direktno zavisi od ponašanja snabdevača. Veza realizacija opisuje korelaciju između specifikacije i realizacije u kojoj je zavisni elemenat implementacija nezavisnog elementa.*

***Ključne reči:** Realizacija, specifikacija, zavisni elemenat.*

# RELATION DEPENDENCY ACCESS

***Summary**: Realization link is one of the dependency links which describe connection between a provider and a client, where client's behaviour directly depends on behaviuor of the provider. Realization link describes correlation between specification and realization in which the dependent element is implementation of independent element.*

***Key words:** Realization, specification, dependent element.*

## 1. UVOD

Vrlo često je potrebno u objektno orijentisanom programiranju gde postoji hijerarhija klasa zajetničke karakteristike izdvojiti u poseban elemenat koji nosi naziv interfejs ili apstraktna klasa. Ovo se radi pre svega da se zajednički elementi ne bi implementirali posebno u svaku klasa već se njihov sadržaj preuzima u implementacionom elementu. Veza koja povezuje specifikacioni element sa implementacionim elementom nosi naziv realizacija.

## 2. VEZA ZAVISNOSTI REALIZACIJA

Ovo je veza između specifikacije i realizacije u kojoj je zavisni element realizacija (implementacija) nezavisnog (stereotip <<realize>>).

Specifikacija uopšte opisuje ponašanje ili strukturu nekog sistema bez određivanja kako će sistem biti implementiran. Implementacija klase prikazuje detalje realizacije njenog ponašanja. Prema tome, postoji veza između jednog elementa koji specificira ponašanje i drugog koji prikazuje implementaciju i nazvana je realizacija. Uopšte, postoji mnogo

[1] Munir Šabanović, dipl. ing. elektrotehnike, asistent na Fakultetu za informatiku i informacione tehnologije, Internacionalnog univerziteta u Novom Pazaru, e-mail: munirsabanovic@yahoo.com
[2] Dr Momčilo Vujičić, vanredni profesor, Tehnički fakultet, Svetog Save 65, Čačak, e-mail: vujicic@tfc.kg.ac.yu, vujicic_momcilo@yahoo.com

načina realizacije iste specifikacije što je opisano narednim listingom u programskom jeziku Java. U ovom listingu specifikacioni elemenat je interfejs I1 a implementacioni elemenat je klasa K.

*Listing 1*: specifikacioni elemenat I1, implementacioni elemenat K

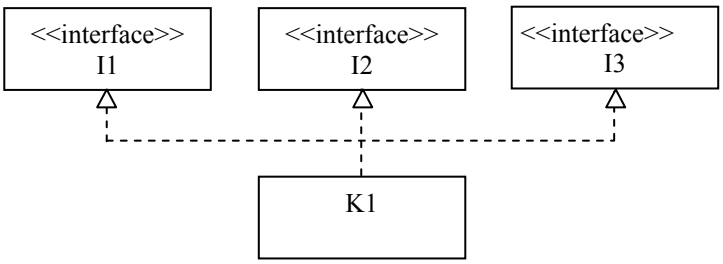
```
public interface I1{
int Inc( );
}

public class K implements I1{
private int i;
public K(int i){ this.i=i; }
int Inc(){
++i;
  }
}
ili

public class K implements I1{
private int i;
public K(int i){this.i=i;}
int Inc(){
i+=1;
  }
}
ili

public class K implements I1{
 private int i;
 public K(int i){this.i=i;}
 int Inc(){
     i=i+1;
   }
}
```

Slično, neki element može da realizuje više od jedne specifikacije kao što je prikazano na slici 1. Na slici 1 specifikacioni elementi su interfejsi I1, I2 i I3 dok je implementacioni elemenat klasa K1.




*Slika 1*

*Listing 2*: reprezentuje vezu na slici 1

```
public interface I1{
…
}
public interface I2{
…
}
public interface I3{
…
}

public class K1  implements I1, I2, I3{
…
}
```

Značenje realizacije je da klijentski elemenat mora implementirati sve metode iz specifikacije. U protivnom, kompajler će prijaviti grešku. Prema tome, zavisni element mora podržati operacije nezavisnog elementa koje obuhvataju sva stanja na koje se može uticati spolja. Znači, implementacioni elemenat mora operacionalizovati sve operacije koje su uključene u specifikacioni element, što je prikazano listingom 3.

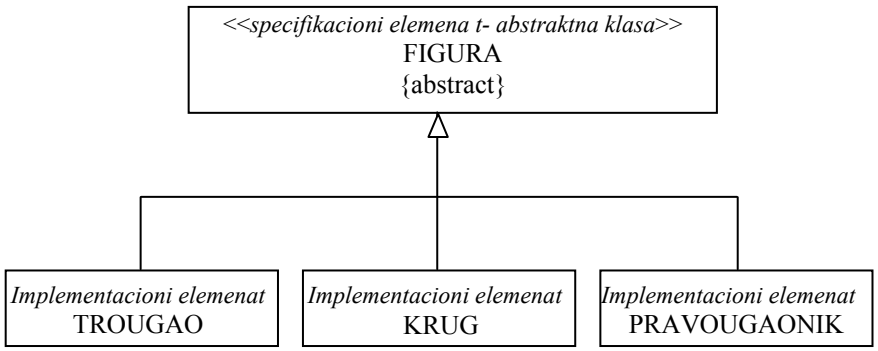
Listing 3:

```
public interface Povrsina_Obim (){          // specifikacioni element Povrsina_Obim
float povrsina ();
float obim ();
}
public interface Zapremina (){              // specifikacioni element Zapremina
float zapremina ();
}
public interface Visina (){                 // specifikacioni element Visina
float visina();
}

public class Krug implements Povrsina_Obim (), Zapremina,Visina{
private             // implementacioni elemenat Krug  operacionalizuje metode iz interfejsa
  float r;
public
  Krug (float r){this.r = r;}
  float povrsina (){return r*r*3.14;}
  float obim (){return 2*r*3.14;}
  float zapremina (){     }
  float visina (){        }
}
```

Dakle, klasa mora da podrži sve operacije iz interfejsa koje se realizuju sa semantikom koja je konsistentna sa celom specifikacijom zahteva u interfejsu. Određena klasa može ostvariti (deklarisati) dodatne operacije, i implementacijom ovih operacija, klasa može da radi dodatne stvari pod uslovom da nisu narušene deklarisane operacije u interfejsu. Neke vrste elemenata kao što su interfejsi odnose se na specifikaciju ponašanja, i one ne sadrže

informacije o implementaciji. Druge vrste elemenata kao što su klase namenjene su za implementaciju ponašanja. One sadrže informacije o implementaciji, ali mogu takođe biti korišćeni na apstraktan način, kao nezavisni elemenat tj. kao apstraktna klasa gde bar jedna metoda nema informacije o implementaciji, što je prikazano na slici 2. Na slici 2 specifikacioni elemenat je apstraktna klasa Figura a implementacioni elementi su klase Trougao, Pravogaonik i Krug, gde immplementacioni elementi podržavaju operacije iz apstraktne klase tj. Iz specifikacionog elementa a veza na slici 2 je opisana narednim listingom.




*Slika 2*

*Listing 4:* reprezentuje vezu na slici 2

```
class Figura {          // specifikacioni element Figura
public:
  Figura() { }
  virtual ~Figura() { }
  virtual double Obim () const=0;
  virtual double Povrsina () const=0;
};
class Trougao:public Figura {        // implementacioni element Trougao
   protected:
      double s1,s2,s3;
  public:
      Trougao (double x,double y,double z):s1(x),s2(y),s3(z) {}
      virtual~Trougao () {}
      double A() const {return s1;}
      double B() const {return s2;}
      double C() const {return s3;}
      double Obim() const {return s1+s2+s3;}
      double Povrsina() const;
};
inline double Trougao::Povrsina() const
{
double p=0.5*(s1+s2+s3);
return sqrt (p*(p-s1)*(p-s2)*(p-s3));
}
```

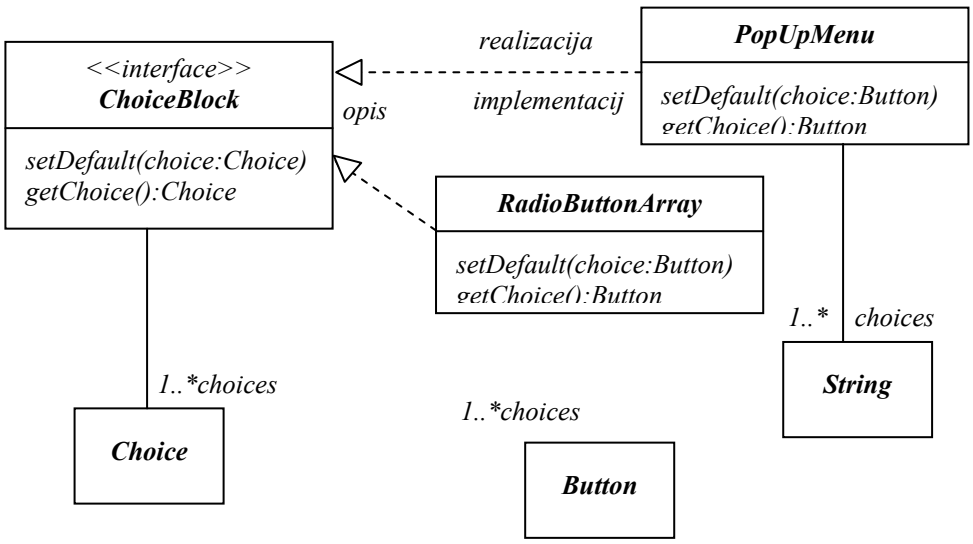
```
class Pravougaonik: public Figura{          // implementacioni element Pravougaonik
protected:
   double s1,s2;
public:
   Pravougaonik(double x,double y):s1(x),s2(y){}
   virtual~pravougaonik(){}
   double a()const{return s1;}
   double b()const{return s2;}
   double obim()const{return 2*(s1+s2);}
   double povrsina() const{return s1*s2;}
};

class Krug:public figura {              // implementacioni element Krug
protected:
  double r;
public:
  Krug(double x):r(x) {}
  virtual~krug()  {}
  double poluprecnik()const{return r;}
  double obim ()const{return2*r*pi;}
  double povrsina()const{return r*r*pi;}
};
```

Realizacija obično povezuje specifikacioni elemenat kao što je interfejs sa implementacionim elementom kao što je klasa. Moguće je i da se koristi implementacioni elemenat kao što je klasa za specifikaciju. Takav elemenat se može tretirati kao specifikacija u vezi realizacije. U ovom slučaju implementacijom samo specifikacionog dela nezavisne klase dobijamo zavisnu klasu. Kako će biti implementirana specifikacija irelevantno je za vezu realizacije. Veza realizacije može da se zamisli i kao nasleđivanje ponašanja specifikacije bez nasleđivanja strukture ili implementacije.

Ako je specifikacioni element apstraktna klasa koja sadrži samo apstraktne metode, bilo kakva specijalizacija apstraktne klase realizuje apstraktnu klasu koja nema ništa sa nasleđivanjem ali ima sa specifikacijom.

Veza realizacije se prikazuje isprekidanom linijom sa zatvorenom trouglastom strelicom na strani nezavisnog elementa i bez posebne oznake na strani zavisnog elementa što je prikazano na slici 3. Na slici 3 data je grafička predstava specifikacionog i implementacionog elementa. Zajedničke karakteristike za klase koje nose naziv **PopUpMenu** i **RadioButtonArray** izdvojene su u specifikacioni elemenat **ChoiceBlock.**




*Slika 3*

## 3. ZAKLJUČAK

Ovde sam proučio vezu realizacija koja se bavi implementacijom specifikacionog elementa. Specifikacioni elemenat može da bude apstraktna klasa ili interfejs. Kako će biti implementirane metode potpuno je irelevantno za specifikacioni elemenat.

## 4. LITERATURA

[1] Dušan Malbaški: *Objekti i objektno programiranje kroz programske jezike C++ i pascal*, 2006.
[2] Booch G.: *Object-Oriented Analysis and Design, second edition*, Addison-Wesley, 1994.
[3] James Rumbaugh, Ivar Jacobson, Grady Booch: *The Unified Modeling Language, Reference Manual*;
[4] Ivana Stanojević, Dušan Surla: *UML, Uvod u objedinjeni jezik modeliranja*
[5] Aho A. V., Sethi R.,Ullman J.D.: *Compilers-Principles, Techniques, and Tools*, Addison-Wesley, Reading, Massachusetts, USA, 1986.
[6] UML Distilled Second Edition A Brief Guide to the Standard Object Modeling Language
[7] Kraus L.:*Programski jezik C++ sa rešenim zadacima,* Mikroknjiga, Beograd, 1994.
[8] Branko Milosavljević, Milan Vidaković: *Java i internet programiranje*, Novi Sad 2002.
[9] http://www.parlezuml.com/tutorials/umlforjava
[10] Dušan Malbaški, *Internet programiranje-deo 1: objekti i objektno programiranje kroz programske jezik Java*, Tehnički fakultet Mihajlo Pupin Zrenjanin, 2007.