TEHNIKA I INFORMATIKA U OBRAZOVANJU

Konferencija
32000 Čačak
9-11. Maja 2008.

UDK: 004 : 371 Stručni rad

VEZA ZAVISNOSTI ACCESS

Momčilo Vujičić [1], Munir Šabanović [2]

Rezime: *U radu je opisana veza zavisnosti access. Ova veza opisuje pod kojim uslovima jedan paket ima pristup elementima drugog paketa. U radu je takođe analizirana i razlika pristupa elementima paketa od strane drugog paketa za vezu access i programski jezik Java. Ova razlika je prezentirana na primeru koji je napisan u programskom jeziku Java. Primer opisuje veze na slici 1.*

Ključne reči: *Paket, pristup, klijentski paket.*

RELATION DEPENDENCY ACCESS

Summary: *Access dependency link is described in the paper. This link describes conditions needed for one package to have access to the other package's elements. The access difference of one package to the other package's elements for access link and Java programming language is also analyzed in the paper. This difference is presented in the example written in Java programming language. An example describes links in the picture number 1.*

Key words: *package, access, client package.*

1. UVOD

Veza access spada u veze zavisnosti, koje imaju zajedničku karakteristiku to što zavisni element modela ne može postojati bez prisustva nezavisnog, (iako to važi i za kompoziciju, [1]). Znači, zavisnost je neka veza između dva elementa gde promena nad jednim elementom (snabdevačem) prouzrokuje promenu nad drugim elementom koji se zove klijent.

2. VEZA ZAVISNOSTI ACCESS

Access je vrsta zavisnosti koja dopušta jednom paketu da se obrati određenim elementima drugog paketa. Inače, „paket" je izraz za modul, koji se koristi u Javi**.** U Paskalu modul se zove „jedinica" (engl. „unit"). Bilo koji paket (klijent) koji se obraća nekom elementu u drugom paketu mora da uveze dotični element koji se nalazi u određenom paketu koristeći

[1] Dr Momčilo Vujičić, vanredni profesor, Tehnički fakultet, Svetog Save 65, Čačak, e-mail: vujicic@tfc.kg.ac.yu, vujicic_momcilo@yahoo.com

[2] Munir Šabanović, dipl. ing. elektrotehnike, asistent na Fakultetu za informatiku i informacione tehnologije, Internacionalnog univerziteta u Novom Pazaru, e-mail: munirsabanovic@yahoo.com

<<access>> ili <<import>> zavisnost u klijentskom paketu, kako bi mogao da koristi usluge nezavisnog paketa. Paket automatski ima slobodan pristup do svih javnih klasa unutar uvezenog paketa.

Neki elemenat u bilo kom paketu ima pristup do svih elemenata koji su vidljivi unutar određenog paketa. Dostupni elementi za vezu access se mogu ukratko predstaviti u nekoliko stavki.

- Bilo koji elemenat definisan unutar nekog paketa je dostupan unutar istog paketa, što važi i za programski jezik Java, osim ako ima specifikator pristupa private.
- Ako je neki elemenat vidljiv unutar nekog paketa, onda je vidljiv unutar svih paketa koji su ugnježđeni unutar istog paketa.
- Ako nekom paketu uvezemo drugi paket, onda svi elementi koji su definisani kao javni u uvezenom paketu biće javni i unutar zavisnog paketa.
- Ako je neki paket naslednik drugog paketa, onda svi elementi koji su definisani kao javni ili zaštićeni u osnovnom paketu vidljivi su unutar naslednika.
- Access i import zavisnosti nisu tranzitivne. Ako A može da vidi B, a B može da vidi C, to nije dovoljno da A može da vidi C.

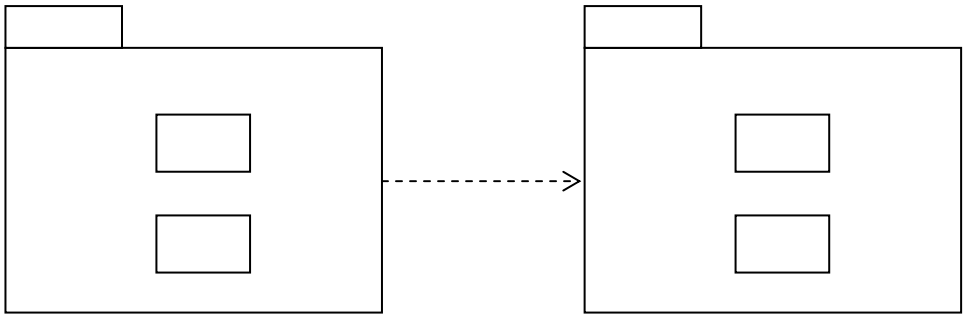
Postoji jedan problem koji se ogleda u tome da paket ne može da „vidi" unutrašnjost sopstvenih ugnježđenih paketa, osim ako je njihov sadržaj *public* i ako ga ne uveze. Međutim ovo pravilo ne važi u Javi, dakle paket „vidi" unutrašnjost sopstvenih ugnježđenih paketa.

Ono što sledi su dalja pravila vidljivosti za klase:

- Delovi klase kao što su podaci-članovi i operacije kao i ugnježđene klase, vidljivi su unutar paketa samo ako je vidljivost tipa default, protected, public elemenata klase.
- Sadržaj klase je vidljiv unutar klase potomka ako ima vidljivost public ili protected u osnovnoj klasi. Ovo važi ako se klase nalaze u različitim paketima. Inače, ako se klase nalaze u istom paketu onda je sadržaj klase vidljiv unutar klase potomka za specifikatore pristupa default, protected i public.

Primer:

Sada ćemo na primeru razmotriti zavisnost tipa *access*. Na slici 1 je prikazan primer razmene podataka između dva paketa. Paket P može da pristupi paketu Q, ali paket Q ne može pristupiti paketu P. Klase K i L u paketu P mogu "videti" javnu klasu M u paketu Q, ali oni ne mogu pristupiti privatnoj klasi N. Klase M i N ne mogu videti nijednu klasu u paketu P, bez obzira na vidljivost tipa public klase K, jer paket Q ne može da pristupi paketu P.



Slika 1

Slika 1 se može opisati narednim listingom u programskom jeziku Java.

```
// kompajlira se sa 'javac -d . -cp . M.java'
package paketi.Q;
public class M {

 public void Introduce(){

        System.out.println("M");
 }
}
// sledeća klasa ne sme biti niti private niti protected - to kompajler
// ne dozvoljava! A ne bi ni imalo smisla jer tad klasi niko ne bi mogao da
// pristupi. Ovako je ona dostupna ostalim klasama iz paketa - "package access"

class N {

 public void Introduce(){
        System.out.println("N");
 }
}
// kompajlira se sa 'javac -d . -cp . K.java' , ali tek nakon M.java

package paketi.P;
import paketi.Q.*;
// sledeća klasa ne sme biti niti private niti protected.
class L {

 public void Introduce(){

        System.out.println("L");
        M Mi = new M();          // ovo može
       // N Ni = new N();        // greška - access nije dozvoljen
        Mi.Introduce();          // ovo može
 }
```

```
}
// samo jedna klasa u fajlu sme i mora da bude public - to je sledeća
public class K {

 public void Introduce(){

        System.out.println("K");
        M Mi = new M();          // ovo može
       // N Ni = new N();        // greška - access nije dozvoljen
        L Li = new L();          // ovo može, L vide svi iz ovog paketa
        Mi.Introduce();
        Li.Introduce();
 }
}
// glavna klasa koja će da poveže sve, tj. da bi moglo nešto da se pozove
// kompajlira se sa  'javac -cp . glavna.java'

import paketi.P.K;

public class glavna {

 public static void main(String args[] ) {

        K Ki = new K();
        Ki.Introduce();
 }
}
```

U Javi zapravo pravi uvoz ne postoji. Ključna reč *import* je tu samo da bi skratila pisanje, tj. da bismo mogli da pišemo samo *M* umesto *Q.M*. *M* je dovoljno jer je prethodno naveden *import Q.**. Takođe, i paket Q može da pristupi svim javnim elementima paketa P: slobodno možemo da definišemo promenljivu tipa *P.K* negde u paketu *Q*, ali to može izazvati komplikacije pri kompajliranju jer se javlja kružna zavisnost (rekurzija, da bismo kompajlirali jedno, treba nam ono drugo i obratno). Originalni Java kompajler može sam da reši ovaj problem.

Prosto rečeno, svim elementima paketa može se pristupiti iz drugih paketa ukoliko su oni javni. Privatnima se ne može pristupiti kao ni onima koji su vidljivi samo članovima paketa (default vidljivost u Javi).

3. ZAKLJUČAK

Ovde sam proučio vezu access koja spada u veze zavisnosti i proučio razlike u pristupima paketa jedan drugom u odnosu na programski jezik Java koji podržava rad sa paketima. Dakle, veza access ne postoji u programskom jeziku Java, a bavi se pristupima elemenata u jednom paketu od strane drugog paketa.

4. LITERATURA

[1] Dušan Malbaški: *Objekti i objektno programiranje kroz programske jezike C++ i pascal*, 2006.
[2] Meyer B.: *Object-Oriented Software Construction*, Prentice Hall, 1988
[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] UML Distilled Second Edition A Brief Guide to the Standard Object Modeling Language
[6] Kraus L.*:Programski jezik C++ sa rešenim zadacima,* Mikroknjiga, Beograd, 1994.
[7] Branko Milosavljević, Milan Vidaković: *Java i internet programiranje*, Novi Sad 2002.
[8] http://www.parlezuml.com/tutorials/umlforjava
[9] Dušan Malbaški, *Internet programiranje-deo 1: objekti i objektno programiranje kroz programske jezik Java*, Tehnički fakultet Mihajlo Pupin Zrenjanin, 2007.