Objektno orijentisano programiranje 2

Paketi

O paketima

- Rešavaju probleme konflikta imena tipova i prijateljstva klasa
- Paketi sadrže tipove (klase i interfejse) i potpakete
- Paketi su korisni iz nekoliko razloga:
  - grupišu logički povezane interfejse i klase
  - članovi paketa (tipovi) mogu da koriste popularna javna imena
  - članovi paketa mogu da budu sakriveni u paketu
- Paketi su donekle slični prostorima imena u jeziku C++
- Definišu se deklaracijama na početku datoteke
- Tipovi iz paketa mogu da se uvoze u datoteke gde se koriste

Deklaracija paketa

- Jedan paket može da obuhvati nekoliko izvornih datoteka
- Svaka izvorna datoteka čiji tipovi pripadaju paketu `X` sadrži deklaraciju:

```
package X;
```

- Deklaracija paketa treba da bude prva linija u datoteci
- U datoteci može da se pojavi samo jedna deklaracija paketa
- Ne postoji “preklapanje” paketa
  - jedan tip je samo u jednom paketu

Uvoz iz paketa

- Postoje dva načina za korišćenje tipova iz nekog paketa:
  - navođenjem punog imena tipa: `<ime paketa>.<ime tipa>`
  - uvozom tipa, pa navođenjem jednostavnog imena tipa
- Uvoz može da bude pojedinačnog tipa ili svih tipova iz paketa
- Primeri:

```
X.A a;          //koristi se klasa A iz paketa X
import X.*;     //uvoze se svi tipovi iz paketa X
import X.A;     //uvozi se samo tip A iz paketa X
A a;            //koristi se uvezena klasa A
```

- Smatra se da paket uvozi sebe implicitno
- Paket `java.lang` se uvozi implicitno

Konflikti imena tipova

- Mehanizam paketa i uvoženja daje kontrolu nad potencijalnim konfliktima imena tipova
- Ako dva paketa (`X` i `Y`) sadrže isto ime `A`, programer koji koristi oba paketa može:
  - da se obraća svim tipovima preko potpunih imena (`X.A`, `Y.A`)
  - da uveze `X.A` i zatim koristi `A` za `X.A`, a potpuno ime za `Y.A`
  - da uradi obrnuto – da uveze `Y.A` i zatim koristi `A` za `Y.A`, a potpuno ime za `X.A`
  - da uveze `X.*` i `Y.*`, pa da koristi potpuna imena (`X.A`, `Y.A`)
- Problem konflikta imena se prebacuje na nivo imena paketa

Imenovanje paketa

- Ime paketa mora da bude jedinstveno
- Naročito veliki problem je sa kratkim, često korišćenim imenima
- Izbor jedinstvenog imena paketa može da se reši na sledeći način:
  - za pakete koji se koriste samo unutar organizacije
    ime treba da se bira koristeći interni arbitar imena
  - za pakete koji se koriste širom sveta, sugestija je:
    koristiti Internet domene inverznim redom
- Primer: `package rs.ac.bg.etf.X`
- Preporuka za razvojna okruženja:
  - da celokupan kod paketa bude u istom folderu (katalogu, fascikli)
  - da se ime foldera koristi za ime paketa

Primer organizacije po folderima

- U osnovnom folderu, fajl `T.java`:

```
import p.*;
public class T{
   public static void main(String[] arg){ A a = new A(); }
}
```

- U folderu `p` koji se nalazi u osnovnom folderu (`.\p`), fajl `A.java`:

```
package p;
public class A{ public A(){System.out.println('A'); } }
```

- Prevođenje:
  - tekući folder je osnovni folder
  - izdaje se komanda: `javac T.java`
  - rezultat prevođenja: `.\T.class`, `.\p\A.class`
- Izvršavanje:
  - tekući folder je osnovni folder
  - izdaje se komanda: `java T`
  - dobija se izlaz: `A`

Uvoz statičkih članova

- Moguće je uvesti i imena statičkih članova neke klase:

```
package x;
public class A {
  public static int sp=100;
  public static void sm() {System.out.println("sm()");}
}

import static x.A.*; // uvezena imena sp i sm
class B{
  public static void main(String[] arg){
    sm();
    System.out.println("sp="+sp);
  }
}
```

Pristup paketu

- Tipovi u paketu mogu da imaju jedno od dva prava pristupa:
  - paketsko i javno
- Tipovi koji nisu javni, podrazumevano imaju paketsko pravo pristupa
  - raspoloživi su za drugi kod u istom paketu
  - sakriveni izvan paketa,
    ne može da im se pristupi čak ni iz ugnežđenih paketa
- Klase u paketu su prijateljske (*friendly*, *trusted*)
  - imaju povlašćen pristup paketskim članovima uzajamno
- Ako je tip `X` javni mora da se definiše u fajlu `X.java`
- Posledica: ne mogu da postoje dva javna tipa u istom fajlu
- Podrazumevano pravo pristupa za članove klasa je paketsko
- Podrazumevano pravo pristupa za članove interfejsa je javno

Hijerarhija paketa

- Tipovi deklarisani u izvornoj datoteci bez deklaracije paketa
  - idu u "neimenovani paket"
- Paketi se mogu da se ugnežđuju u druge pakete
  - može da se napravi hijerarhija (stablo) potpaketa
- Primeri:
  - `java.lang`, `lang` je ugnežđen u `java` paket
  - `java.awt.event`, `event` je potpaket `awt`, koji je potpaket `java`
- Paket `java.lang` sadrži osnovna proširenja jezika (npr. `Object`)
  - podrazumevano se uvozi, nije potrebna deklaracija `import`
- Ugnežđivanje paketa
  - ne proizvodi specifična prava pristupa iz paketa potpaketu ni obrnuto
  - oblast važenja imena iz paketa se ne proteže na ugnežđene pakete
  - u suštini, reč je samo o načinu imenovanja paketa
- Paket `p.sp` može da postoji čak i ako paket `p` ne postoji

Primer paketa

Fajl X.java:

```
package P;
public class X{
    public int xpub=10;
    protected int xprot=20;
    int xpack=30;
    private int xpriv=40;
}
```

Fajl Y.java:

```
package P;
public class Y extends X{
    public int ypub=100;
    protected int yprot=200;
    int ypack=300;
    private int ypriv=400;
    public void printY(){
      System.out.println("printY:");
      System.out.println(xpub);
      System.out.println(xprot);
      System.out.println(xpack);
//    System.out.println(xpriv);
    }
}
```

Fajl Z.java:

```
import P.*;
class Z extends X{
  public void m(){
          System.out.println(xpub);
          System.out.println(xprot);
//        System.out.println(xpack);
//        System.out.println(xpriv);
  }
  public static void main(String[] args){
          Y yo = new Y();
          Z z = new Z();
          z.m();
          System.out.println(yo.ypub);
//        System.out.println(yo.yprot);
//        System.out.println(yo.ypack);
//        System.out.println(yo.ypriv);
          yo.printY();
  }
}
```