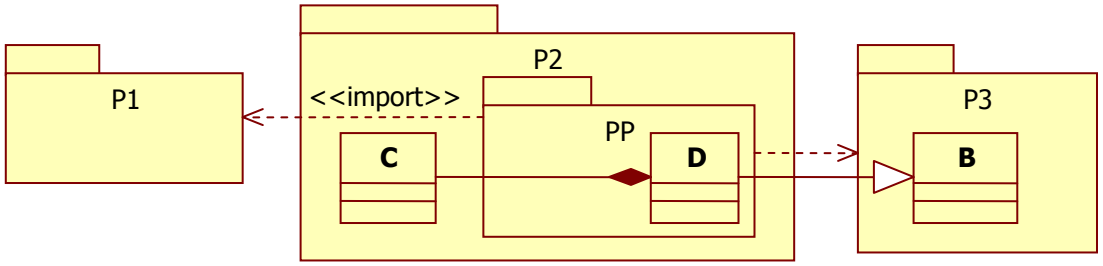
Projektovanje softvera

Dijagrami paketa

Uvod

- Dijagrami paketa se koriste da prikažu dekompoziciju modela u organizacione jedinice i njihove zavisnosti
- Dijagrami paketa pomažu:
  - da se uoče zavisnosti između logičkih celina i
  - da se te zavisnosti drže pod kontrolom
- Primer:



Paket

- Paket je organizaciona stvar koja se koristi za grupisanje elemenata
- Paket predstavlja prostor imena i element koji se može pakovati, tako da se može sadržati u drugim paketima
- Definicija uz UML RM:
  - paket je opšti mehanizam za organizovanje elemenata u grupe, koji uspostavlja vlasništvo nad elementima i obezbeđuje jedinstvenost imena elemenata
- Paketi se koriste
  - uobičajeno za grupisanje logičkih apstrakcija modela (klasa)
  - mogu se koristiti i za grupisanje fizičkih stvari (artefakata ili čak čvorova)
- Paket može da obuhvata druge pakete i proste elemente
  - obično postoji jedan paket u korenu hijerarhije paketa koji predstavlja ceo model
- Koncept paketa donekle odgovara
  - istoimenom konceptu u jeziku Java
  - konceptu prostora imena u jezicima C++ i C#

Imenovanje paketa i elemenata

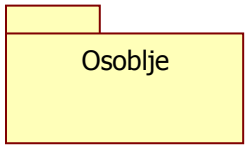
- Jedno ime elementa mora biti jedinstveno u okviru paketa
  - ali se može koristiti za označavanje drugih elemenata iz drugih paketa
- Jednoznačnost imena se odnosi na puna (kvalifikovana) imena
  - puna imena sadrže redom imena svih paketa u hijerarhiji od korenog paketa do datog elementa - lista u stablu, razdvojena simbolom `::`
- Primer:
  - klasa `Panel` koju sadrži potpaket `awt` paketa `java` nosi puno ime `java::awt::Panel`
- Elementi kojima se unutar paketa može obraćati jednostavnim (nekvalifikovanim) imenima su:
  - sopstveni elementi paketa,
  - uvezeni elementi i
  - elementi iz obuhvatajućih (spoljašnjih) prostora imena (paketa)

Vlasništvo i pravo pristupa

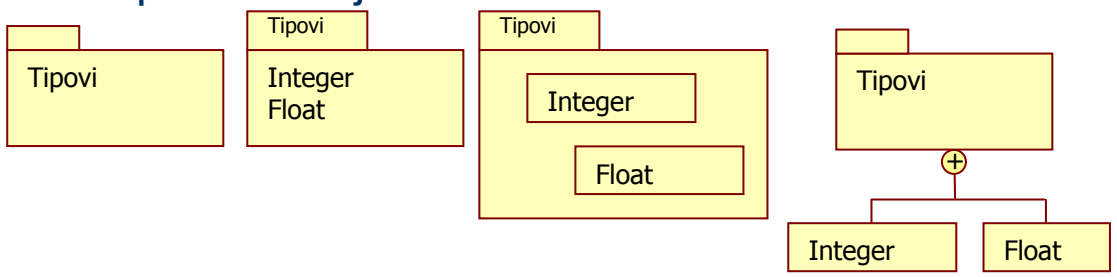
- Vlasništvo nad sopstvenim elementom implicira
  - ako se paket ukloni iz modela, uklanjaju se i sopstveni elementi paketa
  - svaki element modela mora imati kao vlasnika
    - neki paket ili
    - drugi element modela
- Sopstveni i uvezeni elementi mogu imati pravo pristupa
- Pravo pristupa određuje da li su elementi na raspolaganju izvan paketa
- Pravo pristupa elementa u paketu može biti:
  - javno (+) ili
  - privatno (-)
- Javni sadržaj paketa je uvek pristupačan izvan paketa preko punih imena
- Paket "izvozi" svoj javni sadržaj
- Za "uvoz" imena iz drugih paketa koriste se posebne relacije zavisnosti

Notacija

- Za paket se koristi simbol pravougaonika sa jezičkom
  - simbol sugeriše folder koji sadrži:
    - fajlove (jednostavne elemente) i
    - druge foldere (potpakete)

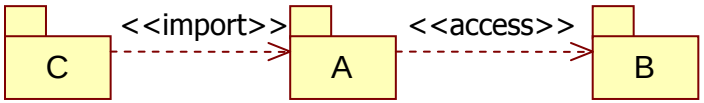


- Sadržani elementi paketa se mogu predstaviti na više načina:
  - samo tekstualno nabrojani unutar pravougaonika simbola paketa (tada se ime paketa piše unutar jezička)
  - nacrtani unutar pravougaonika simbola paketa (i tada se ime paketa piše unutar jezička)
  - povezani punom linijom sa simbolom + unutar kružića na strani paketa



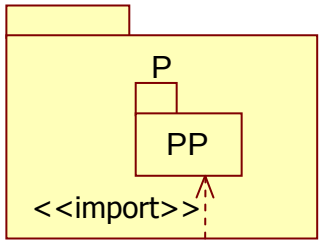
Relacije (1)

- Zavisnosti i stereotipovi zavisnosti: `<<import>>`, `<<access>>`, `<<merge>>`
- Zavisnost:
  - označava da barem jedan element u zavisnom paketu zavisi od nekog elementa iz nezavisnog paketa
  - primer:
    ako je klasa X u paketu P1 izvedena iz klase Y iz paketa P2, paket P1 zavisi od paketa P2
- Javno uvoženje (`<<import>>`):
  - omogućava u paketu u koji se uvozi (na strani repa strelice) korišćenje javnih imena iz uvezenog paketa (na strani glave strelice) bez kvalifikacije
  - uvezeni elementi se ponašaju kao javni u paketu u koji su uvezeni
- Privatno uvoženje (`<<access>>`):
  - omogućava u paketu A u koji se uvozi (na strani repa strelice) korišćenje javnih imena iz uvezenog paketa B (na strani glave strelice) bez kvalifikacije, ali se uvezena imena paketa B ne mogu koristiti bez kvalifikacije u paketu C koji (javno) uvozi imena iz paketa A
  - uvezeni elementi paketa B imaju status privatnih elemenata u paketu A u koji su uveženi



Relacije (2)

- Primeri u jezicima: Java i C# → `<<access>>`, C++ → `<<import>>`
- Alternativna notacija za uvoz pojedinih imena iz drugog paketa je navođenje unutar paketa u koji se uvozi:
  - `{import <kvalifikovano ime>}` ili
  - `{access <kvalifikovano ime>}`
- Uvoz imena sadržanog paketa se ne podrazumeva, a može se naznačiti na sledeći način:
- Može se ispred ili iza ključne reči navesti i alias za uvezeno ime
  - tada originalno ime nije na raspolaganju u paketu
- Mešanje (`<<merge>>`):
  - komplikovana relacija čije korišćenje nije potrebno u modeliranju
  - koristi se u metamodeliranju
- Elementi paketa unutar paketa ili između paketa mogu biti povezani svim vrstama relacija
  - na primer, unutar paketa se može predstaviti klasni dijagram, koji može da sadrži i druge pakete



Principi modeliranja

- Pri projektovanju sistema se nastoji da se broj zavisnosti između paketa minimizira
  - u paket se smeštaju apstrakcije koje uzajamno imaju veći broj relacija i relativno mali broj relacija prema apstrakcijama izvan paketa
- Nije dobro da postoje cirkularne zavisnosti između paketa
  - dobro je da se apstrakcije grupišu slojevito tako da jedan paket predstavlja jedan sloj (nivo) apstrakcija, pa se relacije zavisnosti između paketa orijentišu samo u jednom smeru
- Pravilo zajedničkog zatvaranja (*Common Closure Principle* [R.C. Martin]):
  - preporučuje da u istom paketu budu klase koje treba menjati iz sličnih razloga
- Pravilo zajedničkog ponovnog korišćenja (*Comon Reuse Principle* [R.C. Martin])
  - preporučuje da u istom paketu budu klase koje će se zajedno ponovo koristiti
- Tehnika za sužavanje javnog interfejsa paketa
  - svodi se na izvoženje samo malog broja operacija javnih klasa paketa
  - klase paketa se učine privatnim, a uvede se posebna klasa sa javnim operacijama koje pozivaju odgovarajuće javne operacije privatnih klasa paketa (uzorak *Fasada*)

Stereotipovi paketa

- Iznad imena paketa može stajati naznaka stereotipa << ... >>
- Standardni stereotipovi paketa:
  - koriste se određene ključne reči da označe vrstu paketa
  - model: `<<model>>`
    - semantički potpun opis sistema
    - umesto ključne reči može da se koristi mali trougao
  - radni okvir: `<<framework>>`
    - generička arhitektura kao proširiv obrazac za aplikacije u nekom domenu
    - tipično elementi predstavljaju osnovu za specijalizaciju
- Podsistem `<<subsystem>>` nije stereotip paketa, već komponente
  - u UML-u 1 je smatran stereotipom paketa