

Generarea codului pe baza diagramelor UML de clase

Biblio:

1. Dominik Gessenharter, *Mapping the UML2 Semantics of Associations to a Java Code Generation Model*
2. OMG, *UML 2.5.1 Specification*

Reguli privind reprezentarea asocierilor între clase la nivelul modelului conceptual:

1. Numirea explicită a relațiilor de asociere, folosind expresii verbale sugestive (fac excepție agregările/compunerile, care se numesc implicit prin „este format/compus din”)
2. Precizarea explicită a numelor de roluri (substantive) și a multiplicatilor aferente ambelor capete ale asocierii
3. Conventii: numele de rol se scriu cu litera mica (ca și attributele); dacă multiplicitatea capatului este de tip *many*, se folosește pluralul

Reguli privind traducerea elementelor unei diagrame de clase în cod:

1. Claselor din model le corespund clase în limbajul de programare țintă
2. Mulțimea atributelor unei clase din cod se obține reunind:
 - + mulțimea atributelor clasei sursă din model și
 - + mulțimea numelor de roluri corespunzătoare capetelor opuse ale tuturor asocierilor navigabile dinspre clasa în cauza spre alte clase

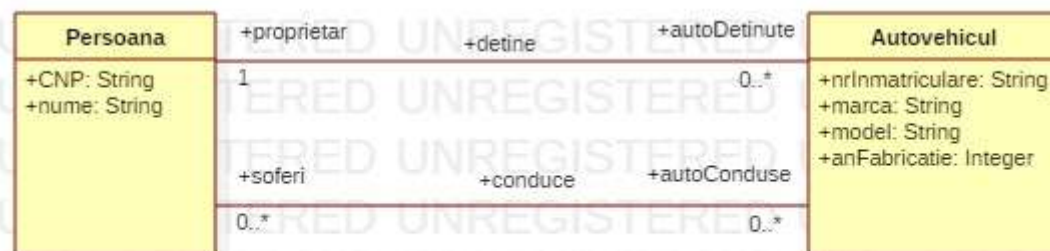
! atenție la conflictele de nume, mai ales în cazul în care o clasă are asocieri diferite cu o aceeași altă clasă, navigabile înspre cea din urmă
3. Tipul unui atribut din cod aferent unui atribut din model e reprezentat de:
 - + corespunzătorul, în limbajul țintă ales, al tipului UML al atributului sursă (ex: Integer -> int, UnlimitedNatural -> long, Real -> double, String -> String, Boolean -> bool) – în cazul în care multiplicitatea aferentă la nivelul modelului este *one* (1 sau 0..1)
 - + colecție, având ca și tip de bază al elementelor tipul menționat anterior, dacă multiplicitatea aferentă la nivelul modelului este de tip *many* (* sau 0..* sau 1..*). Tipul colecției depinde de constrangerile de unicitate și ordonare stabilite (ex.: valorile implicite *unique* = true & *ordered* = false => *Set*)
4. Tipul unui atribut din cod aferent unui nume de rol e reprezentat de:
 - + tipul clasei de la capătul corespunzător rolului, în cazul în care multiplicitatea setată pentru capătul respectiv e *one* (1 sau 0..1)
 - + colecție, având ca și tip de bază al elementelor tipul menționat anterior, dacă multiplicitatea este de tip *many* (* sau 0..* sau 1..*). Tipul colecției depinde de constrangerile de unicitate și ordonare stabilite (ex.: valorile implicite *unique* = true & *ordered* = false => *Set*)
5. La nivelul codului,
 - + unui atribut de tip simplu (primitiv sau referință) îi vor corespunde operații *get/set* (după șablonul *get<numeAtribut>*, *set<numeAtribut>*)

+ unui atribut de tip colecție ii vor corespunde operatii *get/add/remove* (după sablonul *get<numeAtribut>*, *addTo<numeAtribut>*, *removeFrom<numeAtribut>*)

6. ! In cazul asocierilor bidirectionale, cele doua capete ale asocierii trebuie mentinute sincronizate (asociere bidirectionala = doua asocieri unidirectionale + constrangere legată de sincronizarea capetelor) - vezi codul aferent asocierilor bidirectionale din exemplul urmator

Exemplu: Generarea codului pe baza unui model structural UML constand din doua clase (Persoana și Autoturism) și doua relații posibile între acestea (relația de proprietate și cea de utilizare/sofat) – fragment dintr-un model conceptual aferent unui sistem informatic din domeniul asigurarilor auto.

v1: ambele asocieri sunt bidirectionale



v2: prima asociere este unidirectionala, iar a doua bidirectionala

