Seminar 3 **ISS 2019-2020**

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 intre clase la nivelul modelului conceptual:

1. Numirea explicita a relatiilor de asociere, folosind expresii verbale sugestive (fac excepție agregarile/compunerile, care se numesc implicit prin „este format/compus din")

2. Precizarea explicita a numelor de roluri (substantive) și a multiplicitatilor aferente ambelor capete ale asocierii

3. Conventii: numele de rol se scriu cu litera mica (ca și atributele); dacă multiplicitatea capatului este de tip *many*, se foloseste pluralul

Reguli privind translatarea elementelor unei diagrame de clase în cod:

1. Claselor din model le corespund clase în limbajul de programare ținta

2. Mulțimea atributelor unei clase din cod se obtine reunind:
+ mulțimea atributelor clasei sursa din model si
+ mulțimea numelor de roluri corespunzatoare 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 clasa are asocieri diferite cu o aceeași alta clasa, navigabile înspre cea din urma

3. Tipul unui atribut din cod aferent unui atribut din model e reprezentat de:
+ corespndentul, în limbajul ținta ales, al tipului UML al atributului sursa (ex: Integer -> int, UnlimitedNatural -> long, Real -> double, String -> String, Boolean -> bool) – în cazul în care multiplicitatea aferenta la nivelul modelului este *one* (1 sau 0..1)
+ colecție, având ca și tip de baza al elementelor tipul menționat anterior, dacă multiplicitatea aferenta la nivelul modelului este de tip *many* (* sau 0..* sau 1..*). Tipul colectiei 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 corespunzator rolului, în cazul în care multiplicitatea setata pentru capătul respectiv e *one* (1 sau 0..1)
+ colecție, având ca și tip de baza al elementelor tipul menționat anterior, dacă multiplicitatea este de tip *many* (* sau 0..* sau 1..*). Tipul colectiei depinde de constrangerile de unicitate și ordonare stabilite (ex.: valorile implicite *unique* = true & *ordered* = false => *Set*)

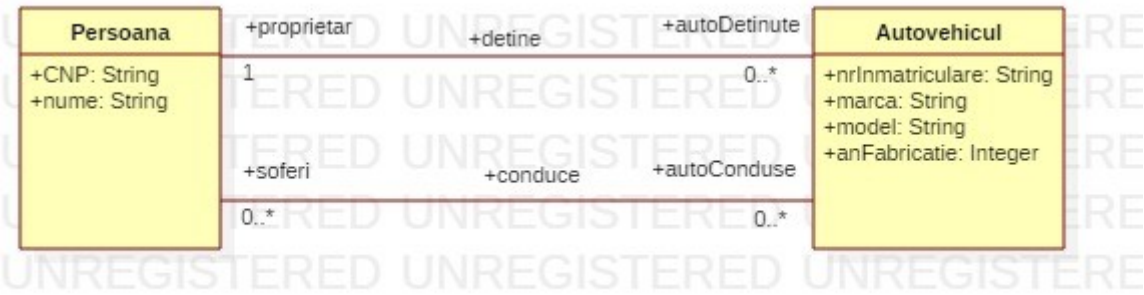
5. La nivelul codului,
+ unui atribut de tip simplu (primitiv sau referinta) ii vor corespunde operatii *get*/*set* (după sablonul *get*<numeAtribut>, *set*<numeAtribut>)

+ 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 intre 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