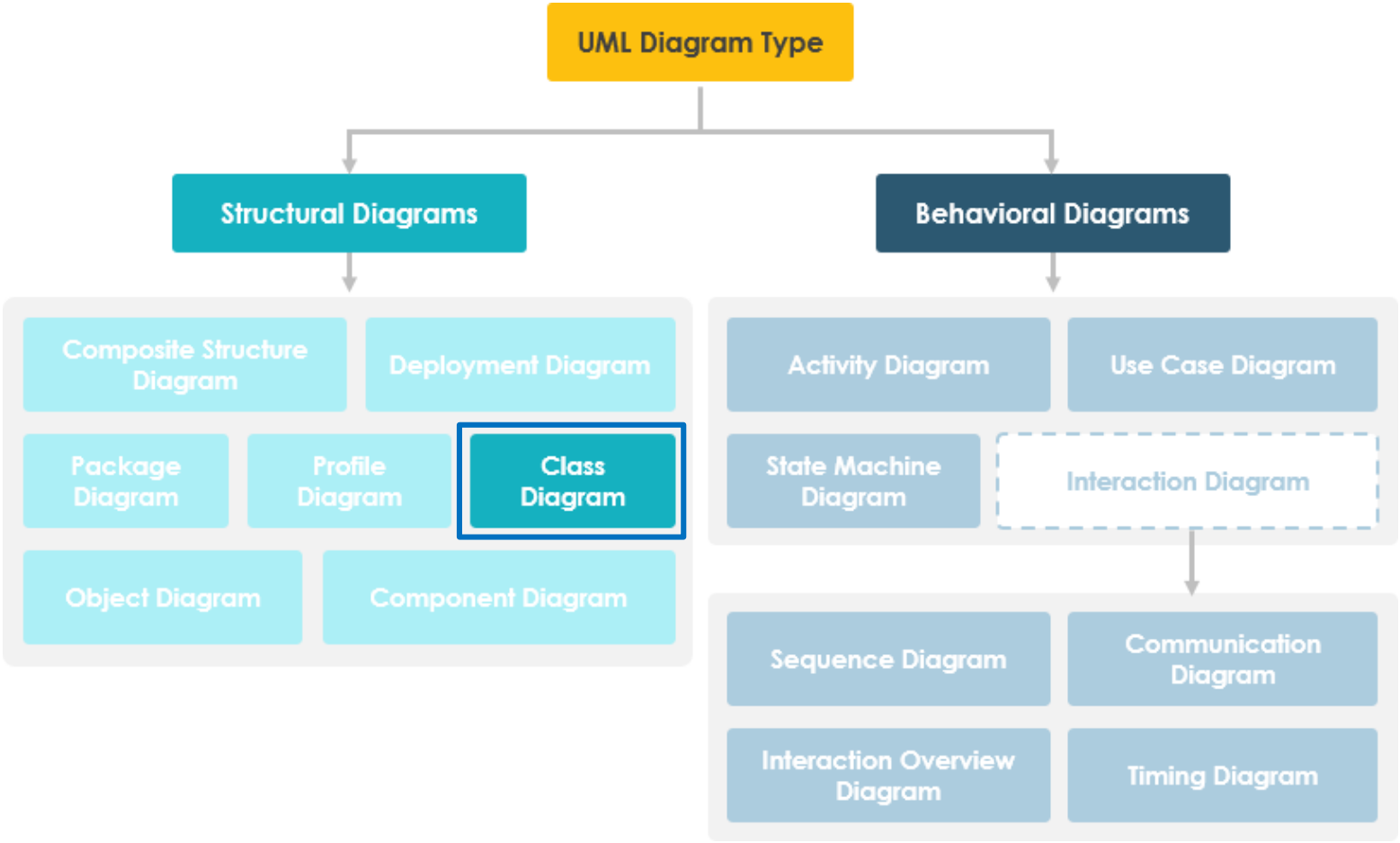
Inxhinieria Sistemeve Softuerike

Modelimi i Sistemit: UML Class Diagrams

FAKULTETI: SHKENCA KOMPJUTERIKE DHE INXHINIERI

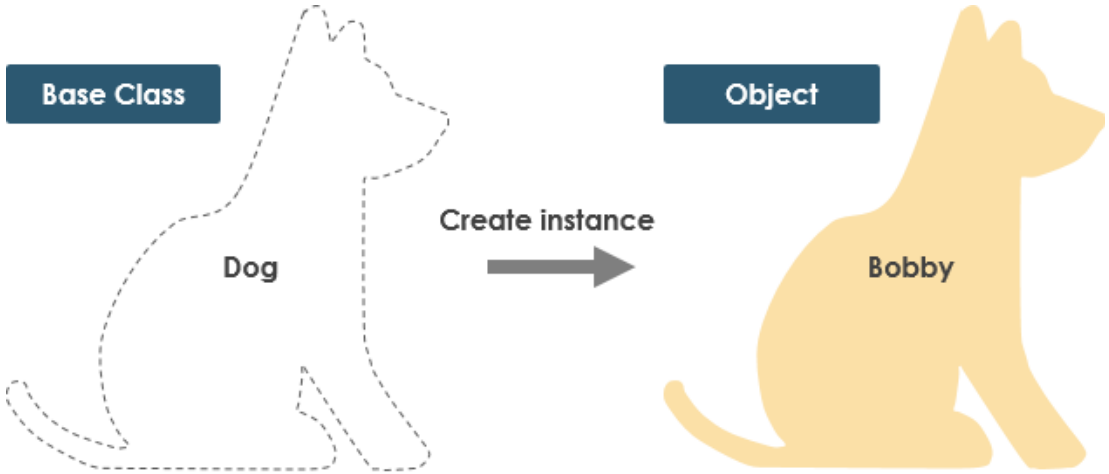
UML definon dy lloj të Diagrameve



Çfarë është një Klasë (1)?

- Një **Klasë** është një *plan* (template) për **objektin**. E gjithë *qëllimi* i **Dizajnit të Orientuar në Objekt (OOD)** nuk ka të *bëjë* me *objektet*, ka të bëjë me **klasat**, sepse ne *përdorim klasa* për të krijuar **objektet**.
- Në fakt, **klasat** përshkruajnë *llojin* e **objekteve**, ndërsa *objektet* janë *raste* të *instancave* të **klasave**.
  - *p.sh*: Një **qen** ka *atributet* - ngjyra, emri, raca, si dhe *sjelljet* – *hece()*, *shkunet()*, *leh()*, *han()*.

Një ***object*** është një instance e **Klasës**



Properties	Methods
Color	Sit
Eye Color	Lay Down
Height	Shake
Length	Come
Weight	

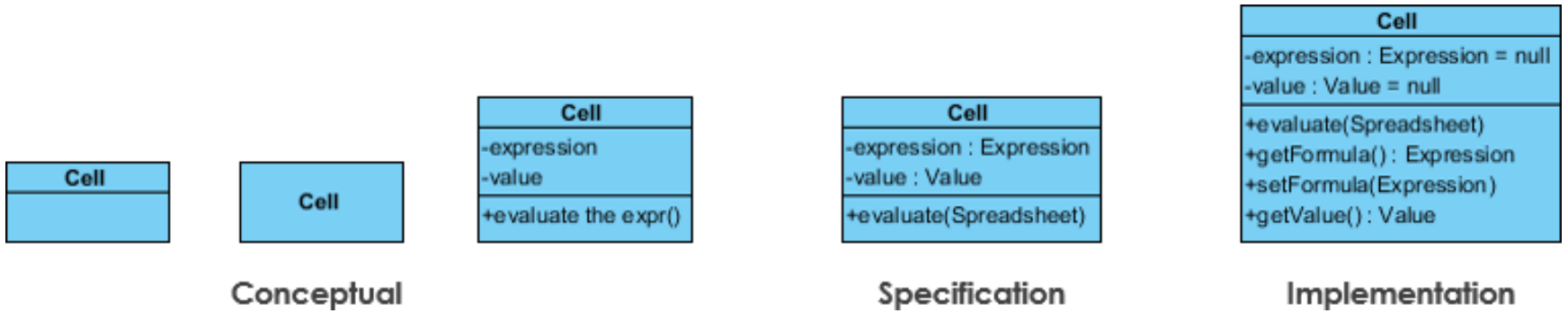
Property Values	Methods
Color: Yellow	Sit
Eye Color: Brown	Lay Down
Height: 17 in	Shake
Length: 35 in	Come
Weight: 24 pounds	

Çfarë është një Klasë (2)

- Një **Klasë** është një ***përshkrim*** i një ***grupi objektesh*** të gjitha me ***role të ngjashme*** në **sistem**, i cili përbëhet nga:
- ***Vetit strukturore (atributeve)*** definojn se çka mund të '**i dim**' për objektet e **klasës**
  - **Reprezentojn** ***gjendjen*** e një objekti të **klasës**
  - Janë ***përshkrime*** të ***veçorive strukturore*** ose ***statike*** të një **klase**
- ***Sjelljes (operacione)*** definojn se çka "**mund të bëjën**" objektet e **klasës**
  - Definojnë ***mënyre*** në të cilen objektet mund të ***ndërveprojn***
  - **Operacione** janë ***përshkrime*** të ***sjelljeve*** ose karakteristika ***dinamike*** të **klasës**

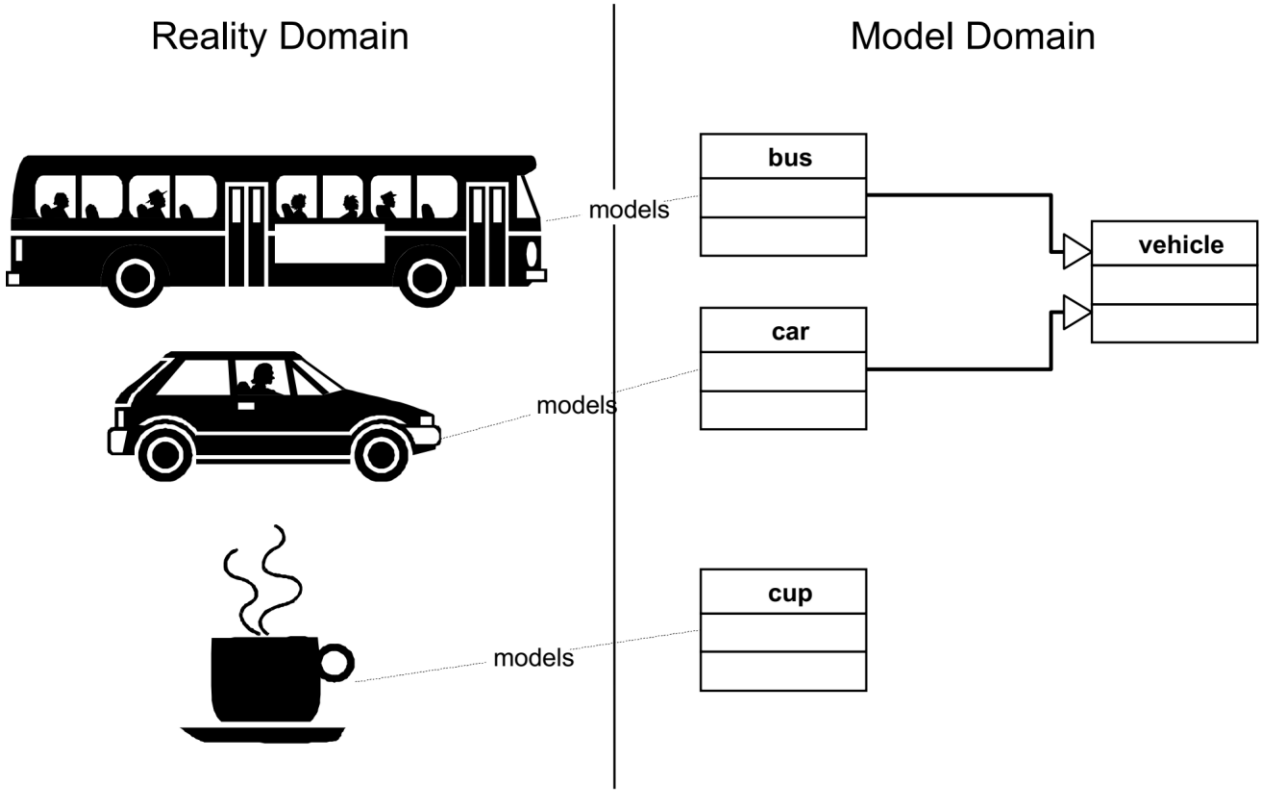
Perspektivat e Diagramit të Klasës në UML

- Një diagram mund të interpretohet nga këndvështrime të ndryshme:
  - *Konceptual*: ***paraqet*** konceptet e **domenit**
  - *Specifikimi*: ***fokusi*** është në **interfaces** e të dhënave abstrakte (ADTs) në softuer.
  - *Implementimi*: ***përshkruan*** se si **klasa** do të ***implementoj*** *interface* e tyre
- ***Perspektiva*** ndikon në ***sasinë*** e ***detajeve*** që duhen furnizuar dhe llojet e *relacioneve* që ia vlen të paraqiten.
  - ***Emri i klasës*** është i vetmi ***informacion i detyrueshëm***.



Object-Oriented Approach
Trajtimi/qasja e orientuar në objekte

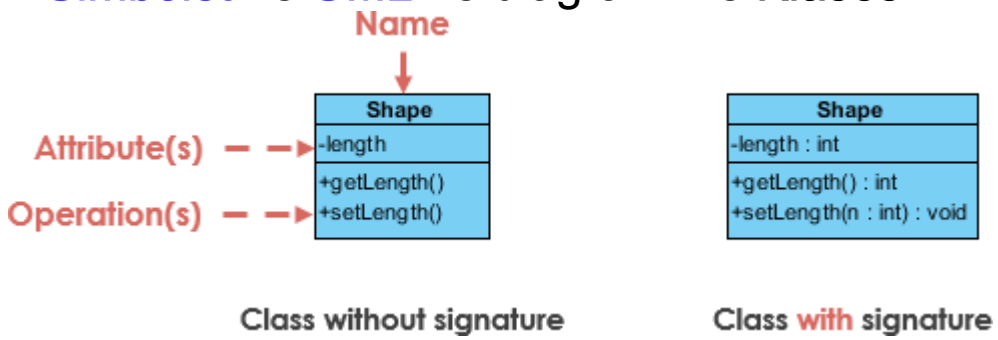
- Objektet janë abstrakte të botës reale ose entitetet të sistemit



Diagrami i Klasës në UML (1)

- Në inxhinieri softuerike, një **diagrami i klasës** në UML (Gjuhë e unifikuar për modelim) është diagrami që paraqet **aspektin statik** që përshkruan **strukturën e sistemit** duke **paraqitur klasat** e sistemit, **atributet** e tij, **operacionet** (ose metodat), dhe **lidhjet/relacionet** ndërmjet **objekteve**.
- **Diagrami i klasëve**, ilustroni modelet e të dhënave për sistemet e informacionit, pa marrë parasysh sa e *thjeshtë* ose *komplekse* janë.
- **Klasat** përfaqsojnë *entitetet e botës reale* ose **koncepte të sistemit**.

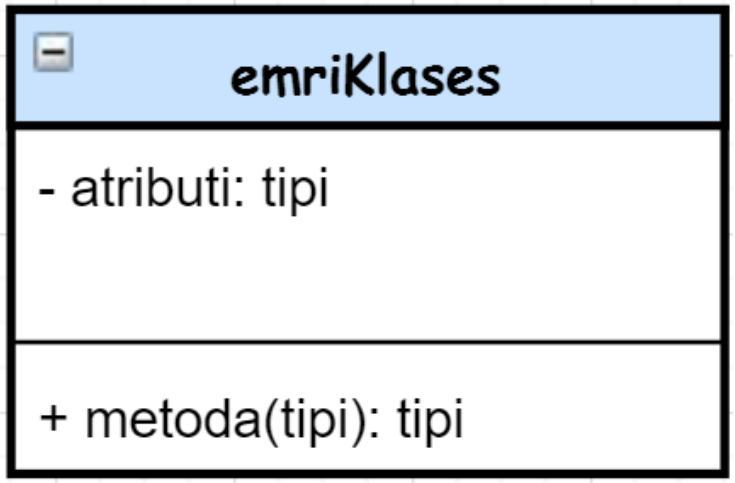
Simbolet në UML në diagramin e **Klasës**



Klasa e objekteve - ***një grup objektesh*** që ndajnë ***atribute*** dhe ***sjellje*** të përbashkëta, nganjëherë i referohemi si një ***klasë***.

Diagrami i Klasës në UML (2)

- Një **klasë** përshkruan ***një grup objektesh*** që kanë:
  - *karakteristika/veti* të **ngjashme** (atributet),
  - *sjellje* të **ngjajshme** (operacione),
  - *relacione* të **përbashkëta** me objekte të tjera, dhe
  - *kuptim* të përbashkët ("semantikë").
- **Karakteristikat** bazë të **UML diagramit të Klasave**:

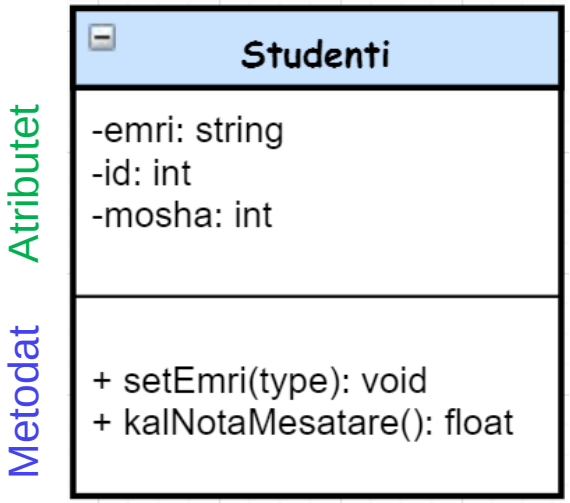


- **Atributet**: një pjesë e rëndësishme e të ***dhënave*** që përmbajnë ***vlera*** që përshkruajnë secilen ***instancë*** të klasës.
- E quajtur edhe ***fusha***, ***variabla***, veti.

- **Metodat**: gjithashtu i quajtur operacion ose funksione.
- Ju lejojnë të specifikoni ndonjë funksion të sjelljes së klasës.

Diagrami i Klasës në UML:
Rasti i kampusi Universitat (Kolegji UBT)

- Rasti i studimit të **kampusi Universitar** (Kolegji UBT).
  - Ne kemi nevoj të i **paraqesimi (reprezentojm)** **gjërat** e ndryshme që **janë në sistem**, këte mund te e **bejme** duke **përdorur** **klasët**.
  - Pra, çfarë ka në kampusin e një universiteti, ka shumë **student**, **profesor**, **departamente**, **puntor administrative**, **objekte/ndërtesat**, **salla labaratorike**, etj.

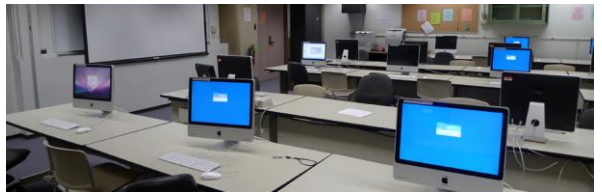


- **Atributet**: një pjesë e rëndësishme e të ***dhënave*** që përmbajnë ***vlera*** që përshkruajnë secilen ***instancë*** të klasës.
- E quajtur edhe ***fusha***, ***variabla***, veti.

- **Metodat**: gjithashtu i quajtur operacion ose funksione.
- Ju lejojnë të specifikoni ndonjë funksion të sjelljes së klasës.

studenti

salla labaratorike



puntori administrativ

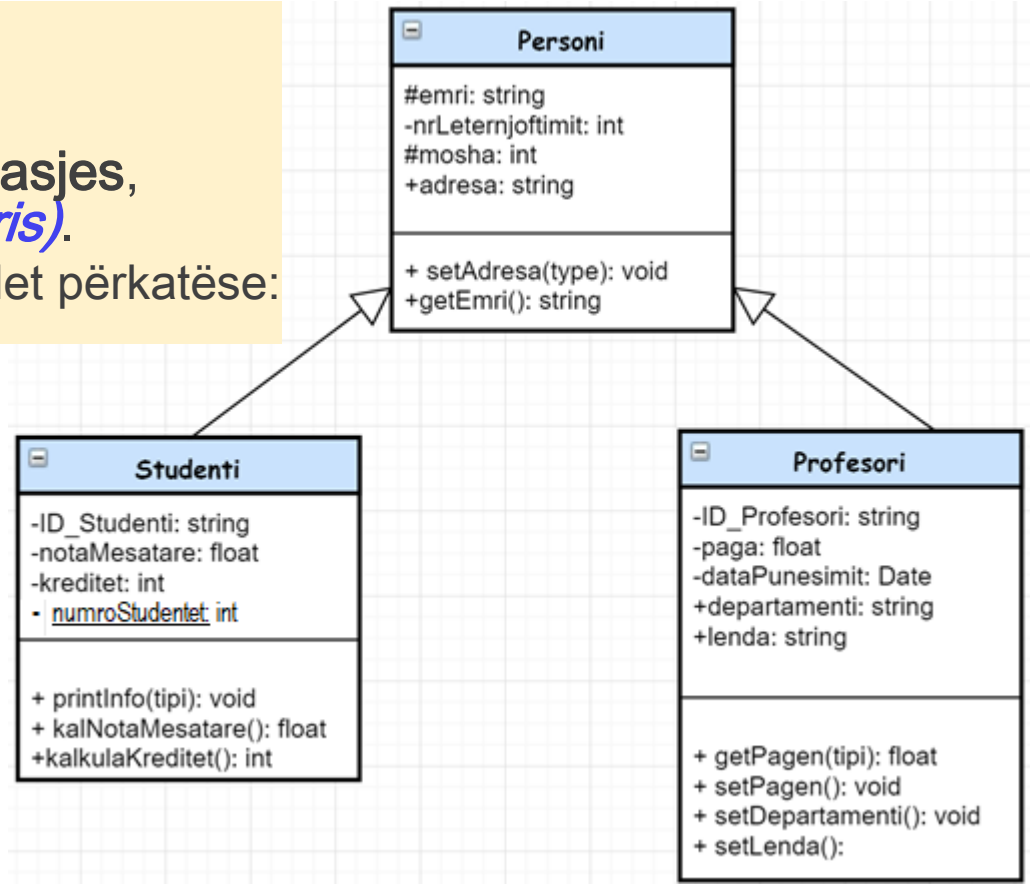
Profesori

Ndërtesat/objektet

Diagramet e Klases në UML: ***Specifikusit e qasjes (1)***

- Specifikuesit e **qasjes** (dukshmëria) së anëtarëve/variableve.
- Të gjitha klasat kanë nivele të ndryshme të **qasjes**, varësisht nga ***modifikuesi i qasjes (dukshmëris)***.
  - ✓Kemi nivelet e mëposhtme të ***qasjes*** me simbolet përkatëse:

Qasja në Klasë (OOP)	UML simbole
public	+
private	-
protected	#
packege	~
static	nënvizuar

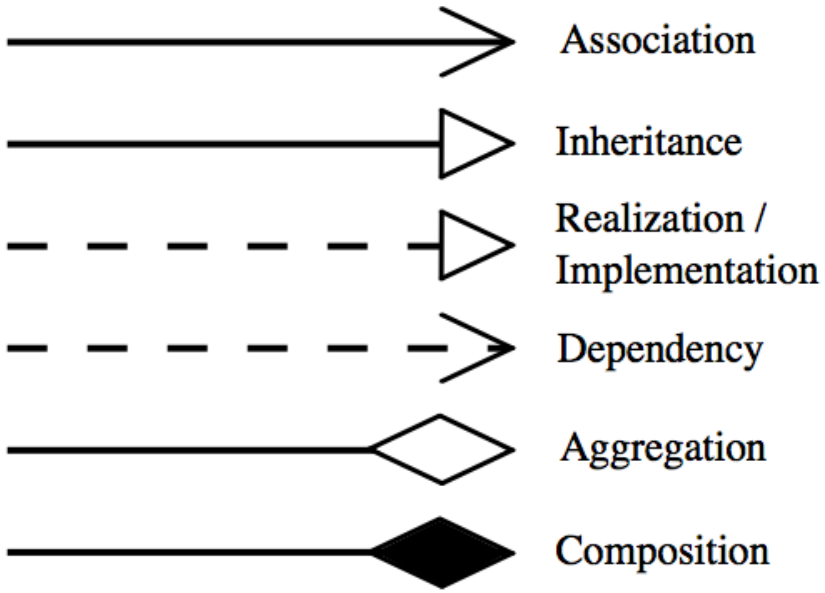


Diagramet e Klases në UML: *Specifikusit e qasjes (2)*

Qasja në Klasë (OOP)	UML simbolet	
public	+	Anëtarët (***atributet***) e të dhënave publike janë të **qasshëm** *brenda* klasës, *jashtë* klasës, nga *çdokush*
private	-	Anëtarët (***atributet***) e të dhënave private janë të **qasshëm** *vetëm brenda* klasës në të cilën janë dëfinuar
protected	#	Variablat e mbrojtura (protected) janë atributet të qasshëm *brenda* klasës në të cilën janë *definuar* dhe nga të gjitha klasat e *fëmijëve* të klasës
packege	~	Anëtarët e të dhënave package mund të janë të qasshëm *brenda* klasës në të cilën janë *definuar* dhe gjithashtu *brenda të gjitha* klasave në atë *paketë*
static	nënvizuar	Anëtarët e të dhënave statike janë atributet që *shfrytzohet (ndahen)* nga të *gjithë objektet e klasës*. Ekziston vetëm një kopje e anëtarëve të të dhënave statike

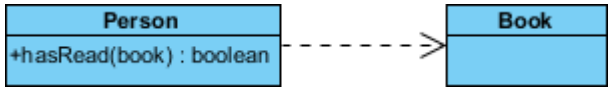
Diagramet e Klasës në UML: Relacionet

- Një relacion është një term i përgjithshëm që mbulon llojet e veçanta të lidhjeve logjike të gjetura në diagramet e klasës dhe objekteve.
- UML përcakton relacionet e mëposhtme:

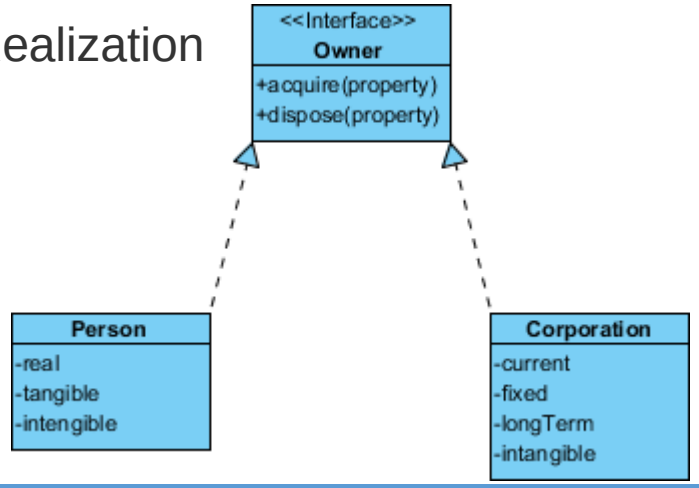


Diagramet e Klasës në UML: Relacionet (2)

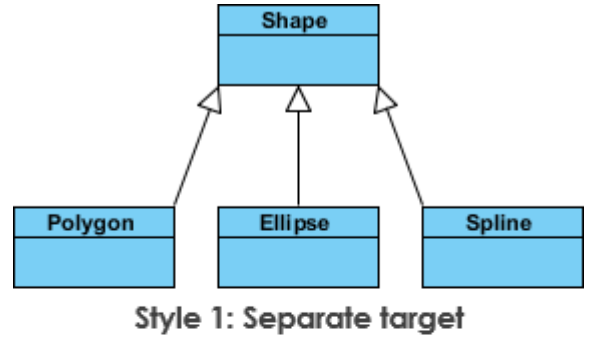
Dependency



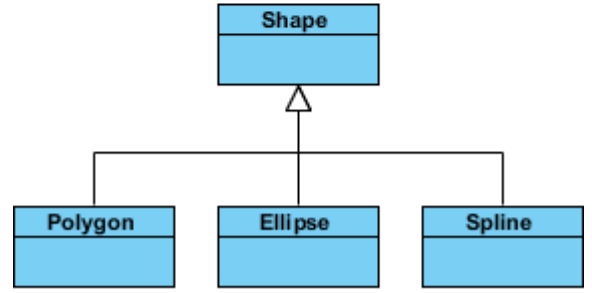
Realization



Inheritance



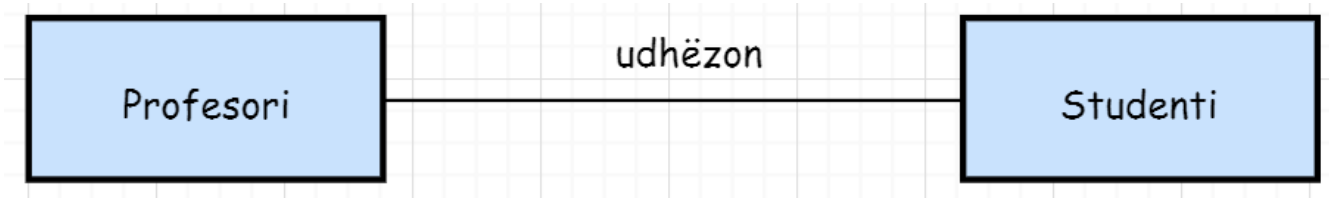
Style 1: Separate target



Style 2: Shared target

Diagramet e Klasës në UML: **Asociacionet**

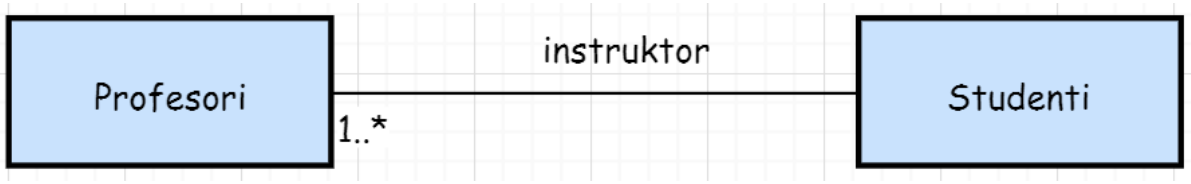
- Nëse dy klasa në një model duhet të *komunikojnë* me *njëri-tjetrin*, duhet të ketë lidhje midis tyre. Një asociacion tregon atë lidhje.
- **Asocimi** ndërmjet dy klasave *tregon si objektet* në një anë të një asociacioni "***njohin***" **objekte** në anën tjetër dhe mund të dërgojnë mesazhe tek ata.
  - Nëse një asocim është drejtuar, mesazhet mund të kalojnë **vetëm në atë drejtim**
  - Nëse asocimi nuk ka drejtim(direkcion), atëherë lidhja është **definuar** si një **lidhje dydrejtimesh** dhe mesazhet mund të kalojnë në të dy drejtimet
  - Sipas paracaktimit (**default**), të gjitha relacionet duhet të drejtohen, përveç nëse kërkesat kërkojnë lidhje dydrejtimesh.



këtu, një asociacion/lidhje është ***udhëzon***.

Diagramet e Klasës në UML:
Asociacionet & pjesamarrja

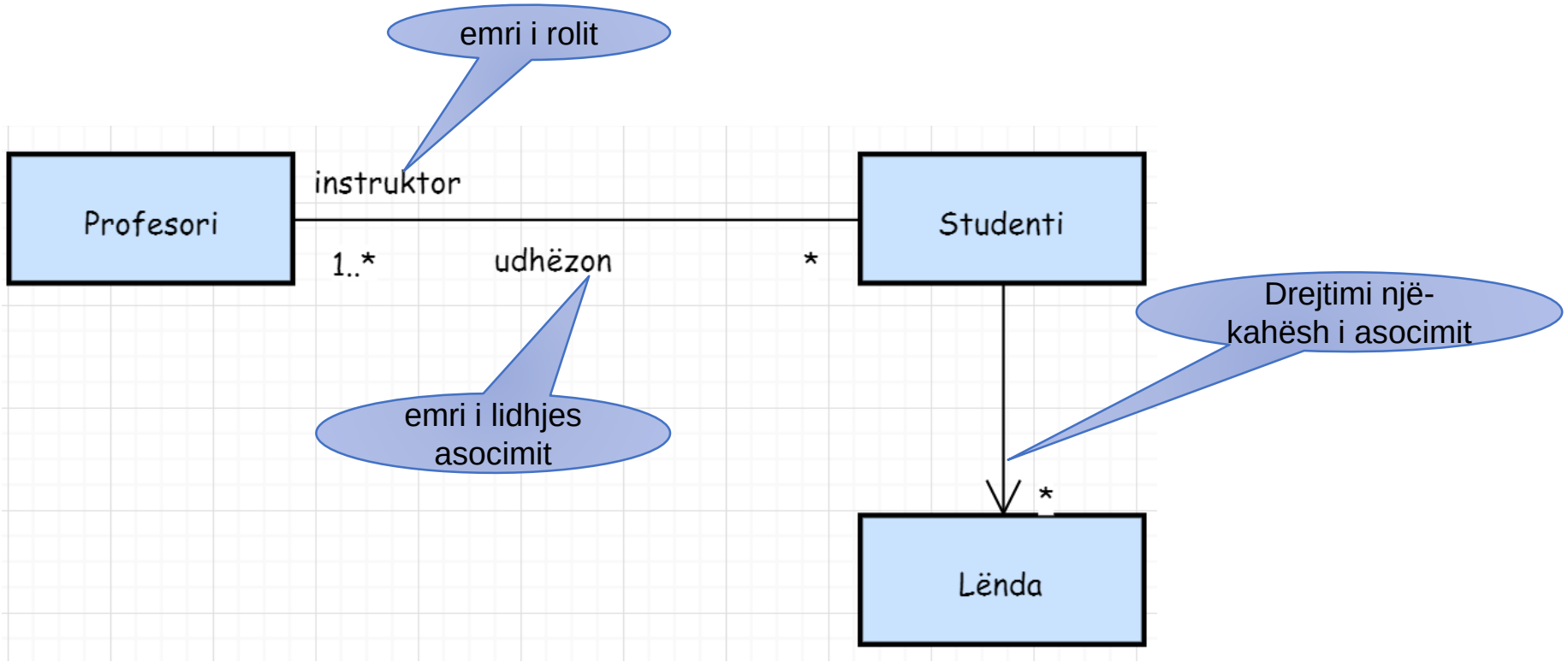
- Ne mund të tregojmë pjesmarrjen e një asocimi duke shtuar pjesmarjen e shumëfishta në vijën që tregon lidhjen.
  - *Numrat prane* **klasave** tregojne se sa *është pjesmarrja* ne këtë lidhje.
  - Shembulli tregon kur **një student** ka **një** ose më **shumë instruktorë**:



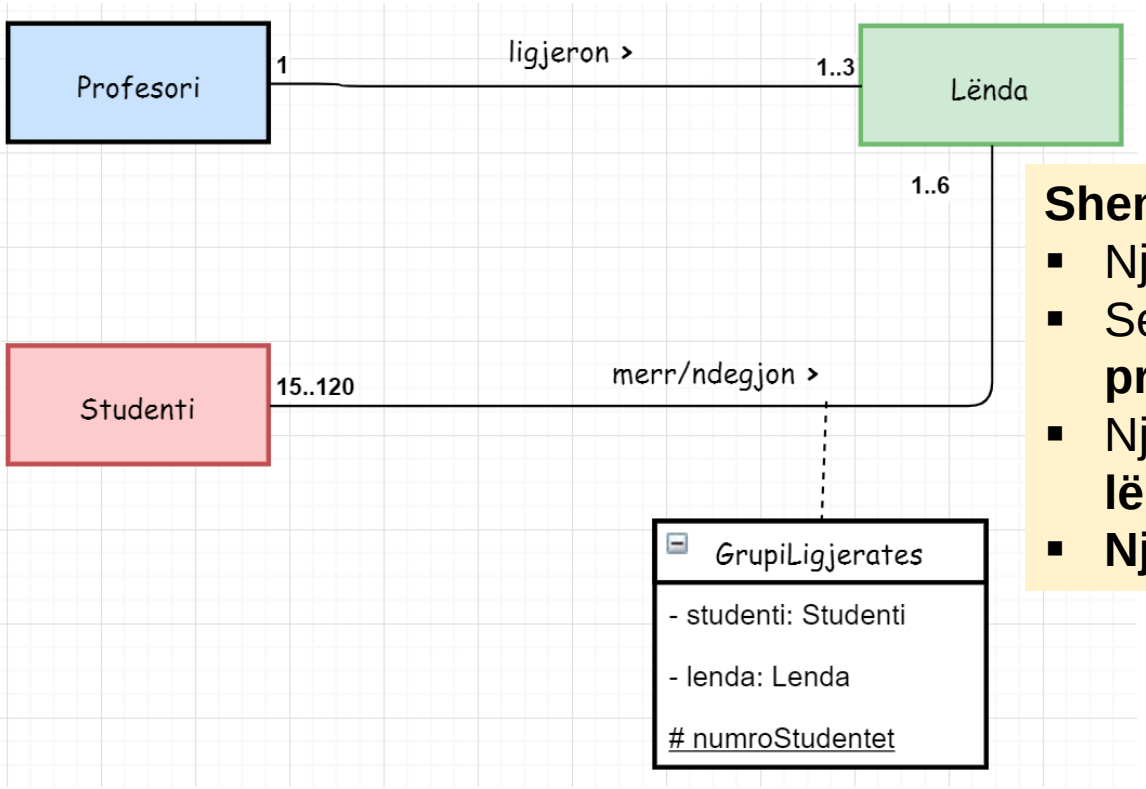
Vlera e tregusit në pjesmarrje	Simbolet e vlerave te pjesmarrjes
0	zero/asnje instance
1	një instance të vetëm
*****	Vlere e pakufizuar int jo negative e instancës
0..1	zero 0 ose një instance të vetme
0..*	zero 0 ose numer të pakufizuar (shume) të instancës
1..*	një ose më shumë instanca
3..6	Rang i specifikuar

Diagramet e Klasës në UML:
Asociacionet & pjesamarrja

- Shembulli 1: Tregon kur min *një profesor* ose më shumë *udhëzon/udhëzojnë* shumë **student**, po ashtu **studentët** kan shumë **lëndë** për të i ndegjuar.

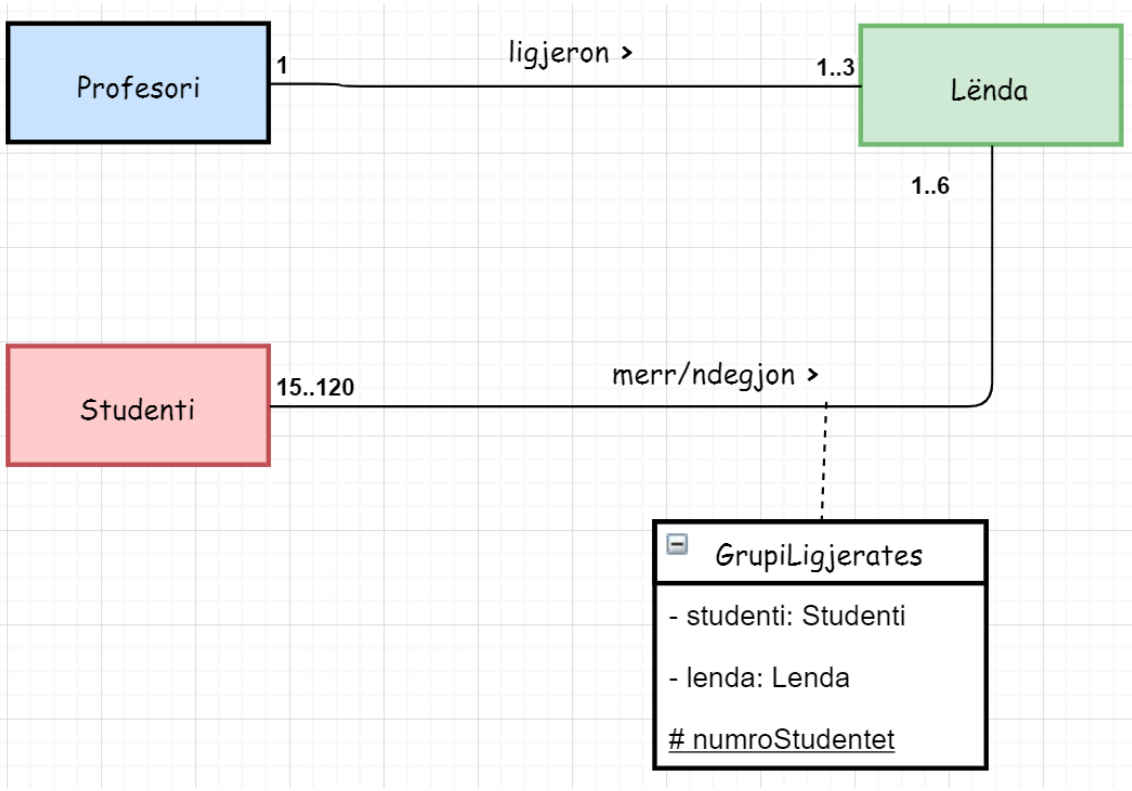


Diagramet e Klasës në UML:
Asociacionet & pjesamarrja



Shembulli 2:

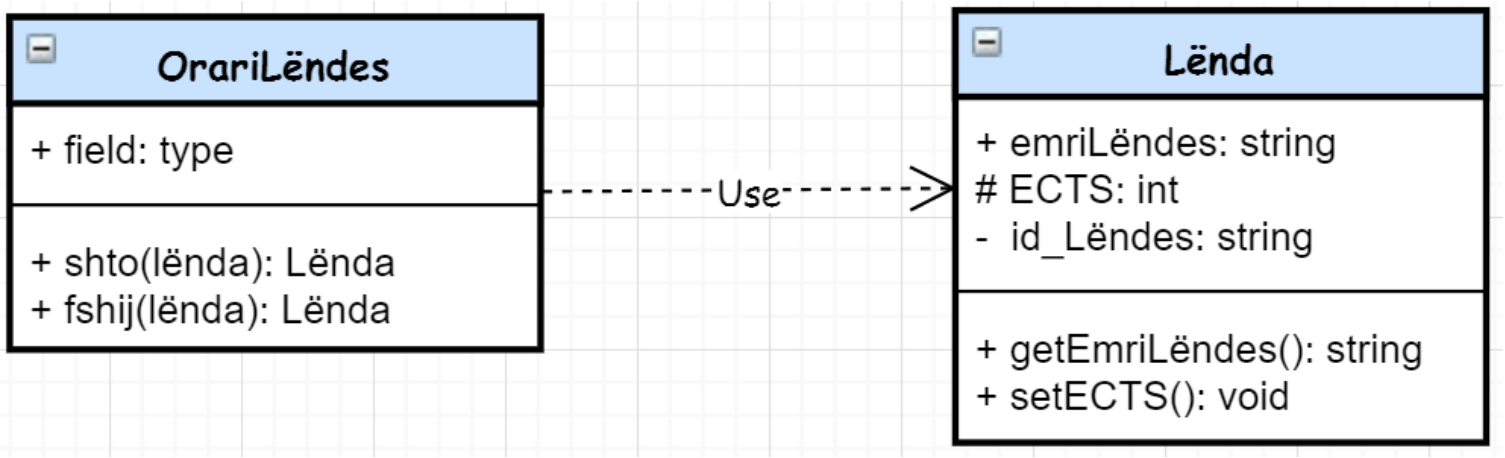
- Një **profesor** *ligjeron* 1 deri në 3 **lëndë**
- Secila **lënd** *ligjerohet* nga **vetëm një profesor**.
- Një **student** mund të *merr/ndegjojë* 1-6 **lëndë**.
- **Një lëndë** mund të ketë 15-120 **student**.




```
public class Profesori {
private String ID_Profesori;
Private String emri;
private float paga;
}
public class Studenti {
private String ID_Studenti;
Private String emri;
private  int kredit;
}

public class Lenda {
    private String emriLendes;
    private Profesori ligjerusi;
}

public class GrupiLigjerates {
    private Lenda[] lende;
    private Studenti[] student;
    private Lenda[] profesori;

    ArrayList<Lenda> lende = new ArrayList<Lenda>();
    ArrayList<Studenti> Student = new ArrayList<Studenti>();

    public GrupiLigjerates(){
        lende = new Lenda[6];
        student = new Studenti[120];

        ……
    }
}
```

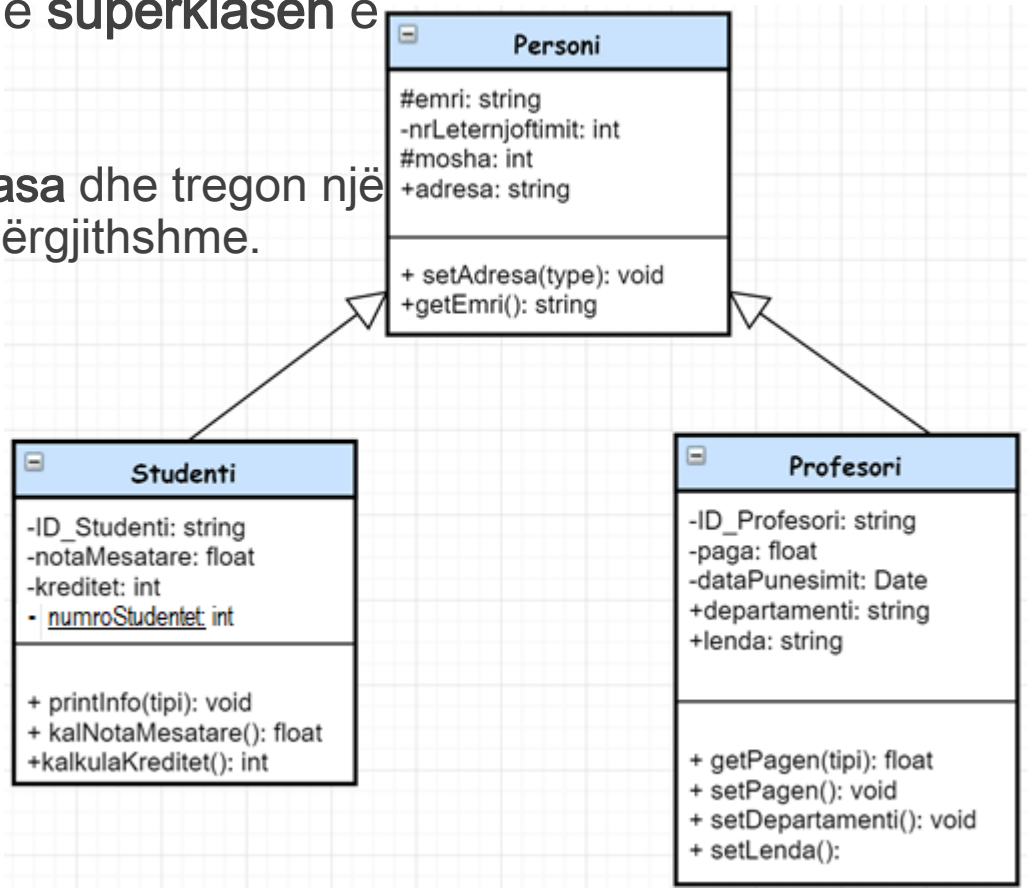
Diagramet e Klasës në UML:
Relacioni i **varësisë (Dependency)**

- Një relacioni i *varësiës* tregon një relacion semantike midis **dy ose më shumë elementeve**.
- Vartësia nga **OrariLëndes** në **Lënda** ekziston sepse **Lënda** *përdoret* edhe për operacionin futjes/shtimit dhe fshirjen/heqjes të orarit të **lëndes** (OrariLëndes).

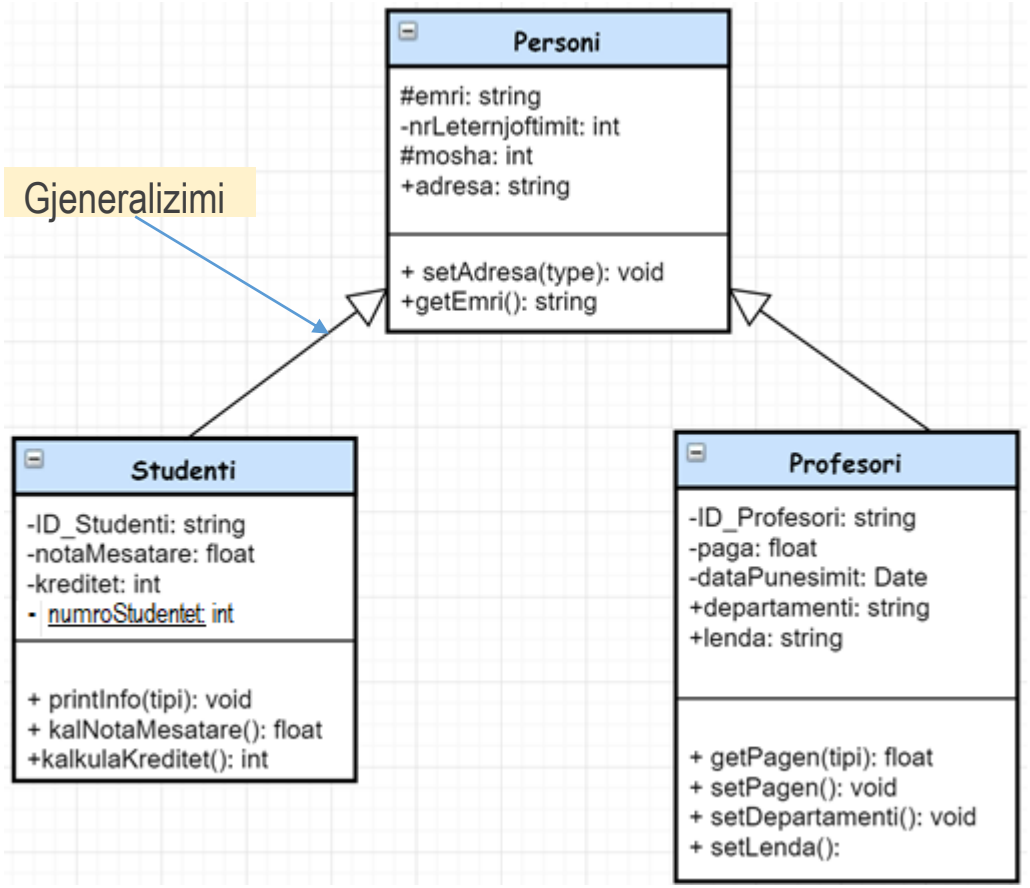


Diagramet e Klasës në UML: Relacioni i **gjeneralizimi/trashigimia**

- Një relacioni i **gjenralizimit** të një **nënklasë** me **superklasën** e saj.
  - Kjo nënkupton një ***trashëgim*** të atributeve dhe sjelljeve/operacioneve nga **superclass** në **nënklasa** dhe tregon një specializim në nënklasën e superklasës më të përgjithshme.



Relacioni i **gjeneralizimi/trashigimia**

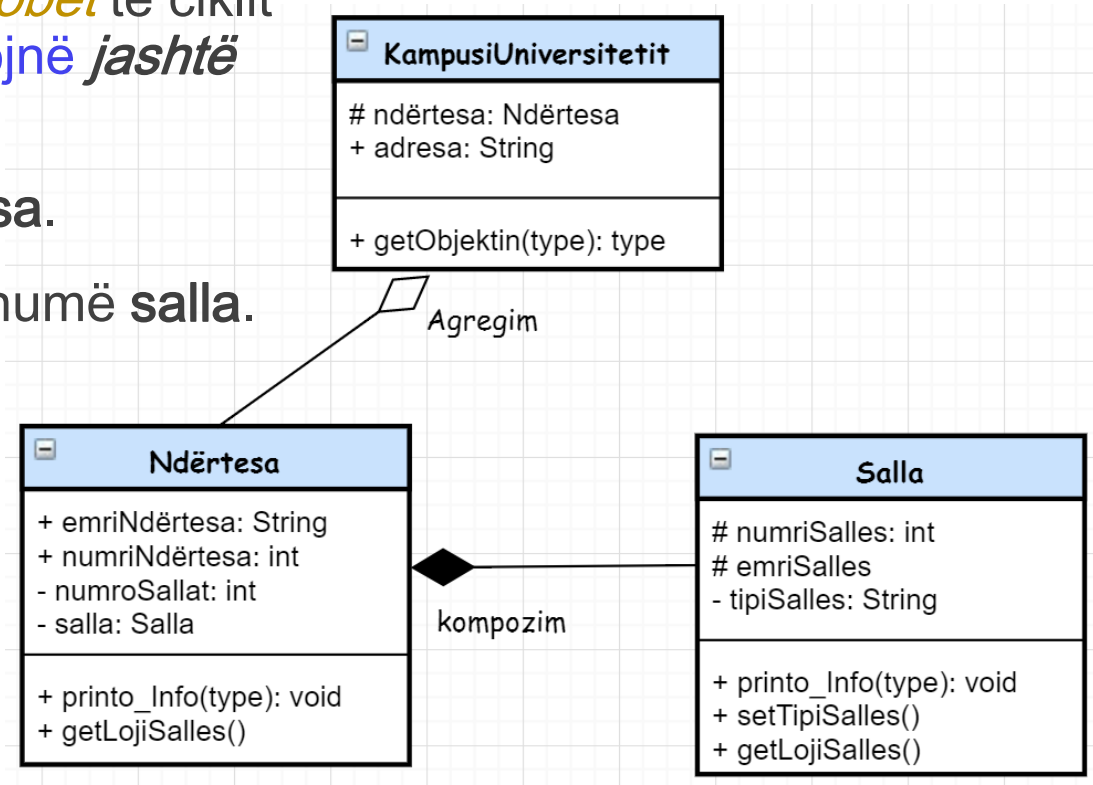



```
public class Personi {
    protected String emri;
    private int nrLeternjoftimit;
    protected int mosha;
    public String adresa;


    public void setAdresa(String adresa) {
        this.adresa = adresa;
    }
    public String getEmri() {
        return emri;
    }
}
public class Studenti extends Personi {
    private String ID_Studenti;
    private float notaMesatare;
    private int kreditet;
    private static int numroStudentet;

    public void printInfo() {
        System.out.println();
    }
    public float getkalNotaMesatare() {
        return notaMesatare;
    }
    public void setInfo(String IDStudenti, String emri, float nm ) {
        this.ID_Studenti = IDStudenti;
         this.emri = emri;
         this. notaMesatare= nm;
    }
}
```

Diagramet e Klasës në UML: Agregimi & Kompozimi

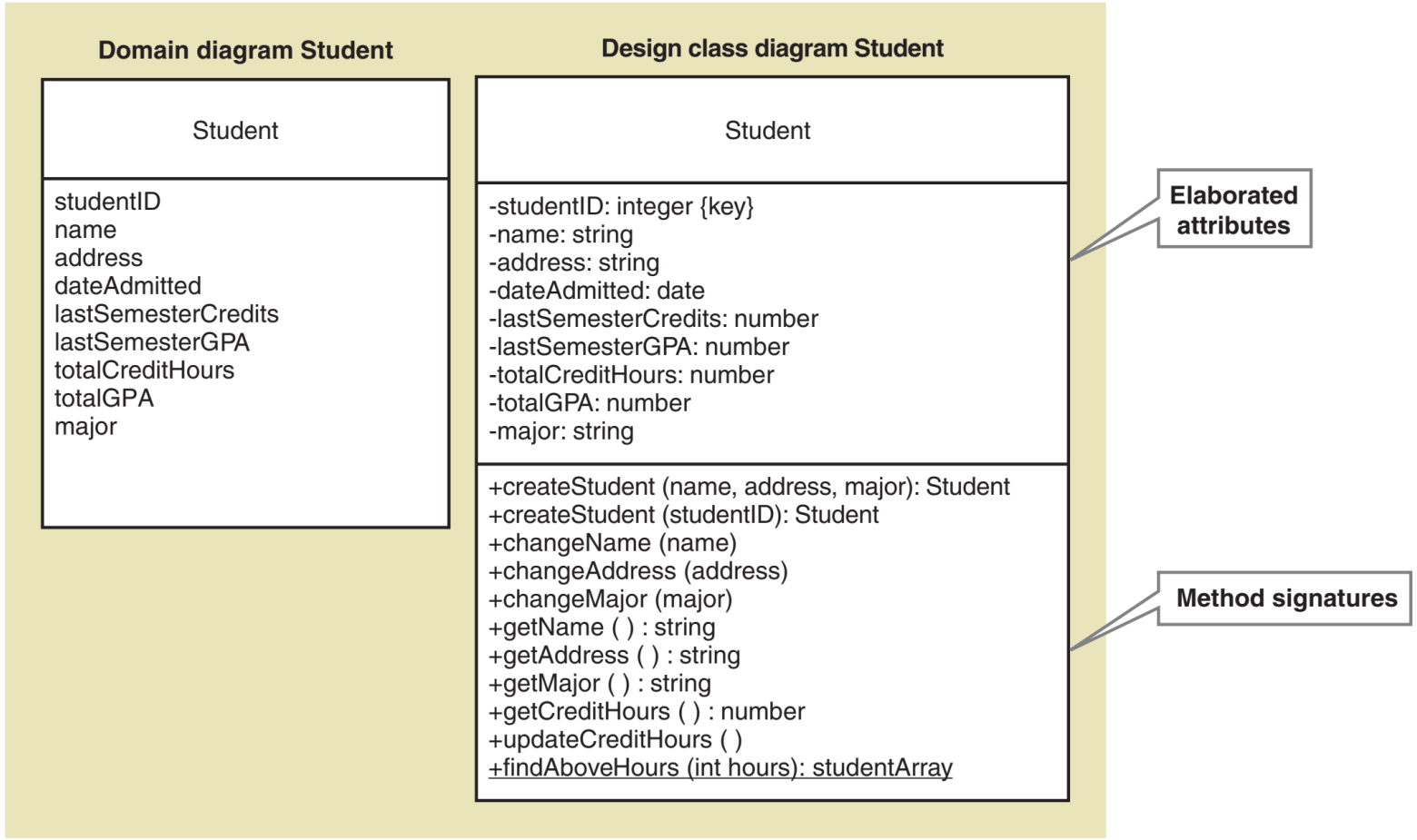
- Përdorni **agregimin** kur ka një *varësi të dobët* të ciklit të jetës, d.m.th. **Ndërtesat** mund të ekzistojnë *jashtë* një *Kampusi univesitar*.
- Një **kampus** përbëhet nga shumë **ndërtesa**.
- Përderisa një **ndërtesë** është bërë nga shumë **salla**.

```
public class Salla {
}
public class Ndertesa {
     private Salla[] salla;
}
public class
KampusiUniversitetit{
     private Ndertesa ndertesa;
}
```



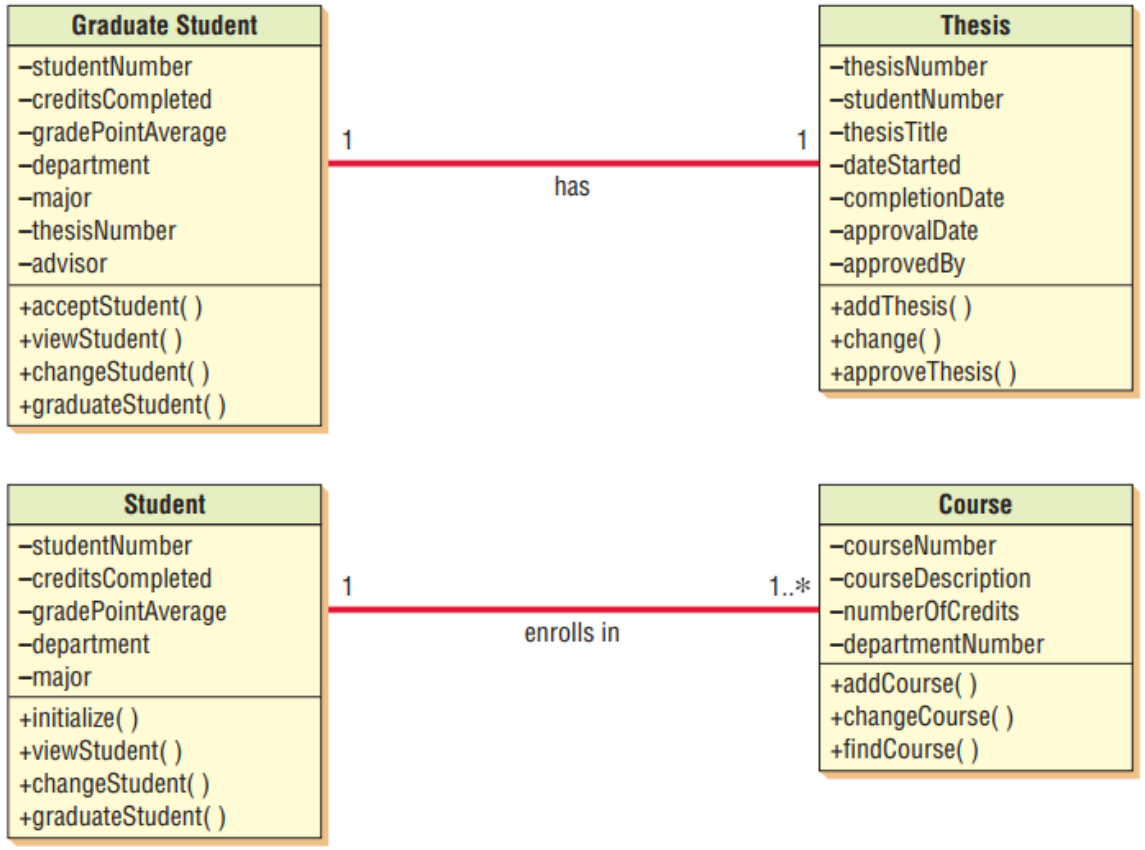
Shembull të diagramit të Klasës

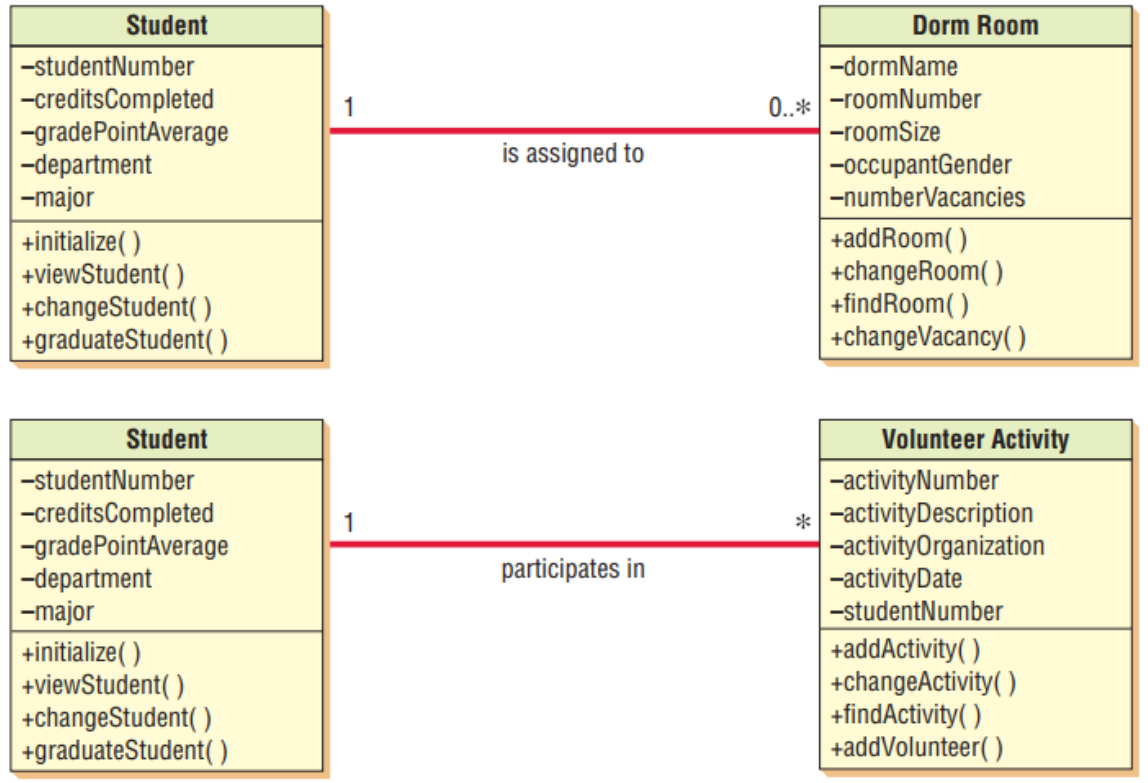
shembuj: të klasës së studentëve
të klasën e ***domenit*** dhe të diagramet i ***dizajnit të klasës***



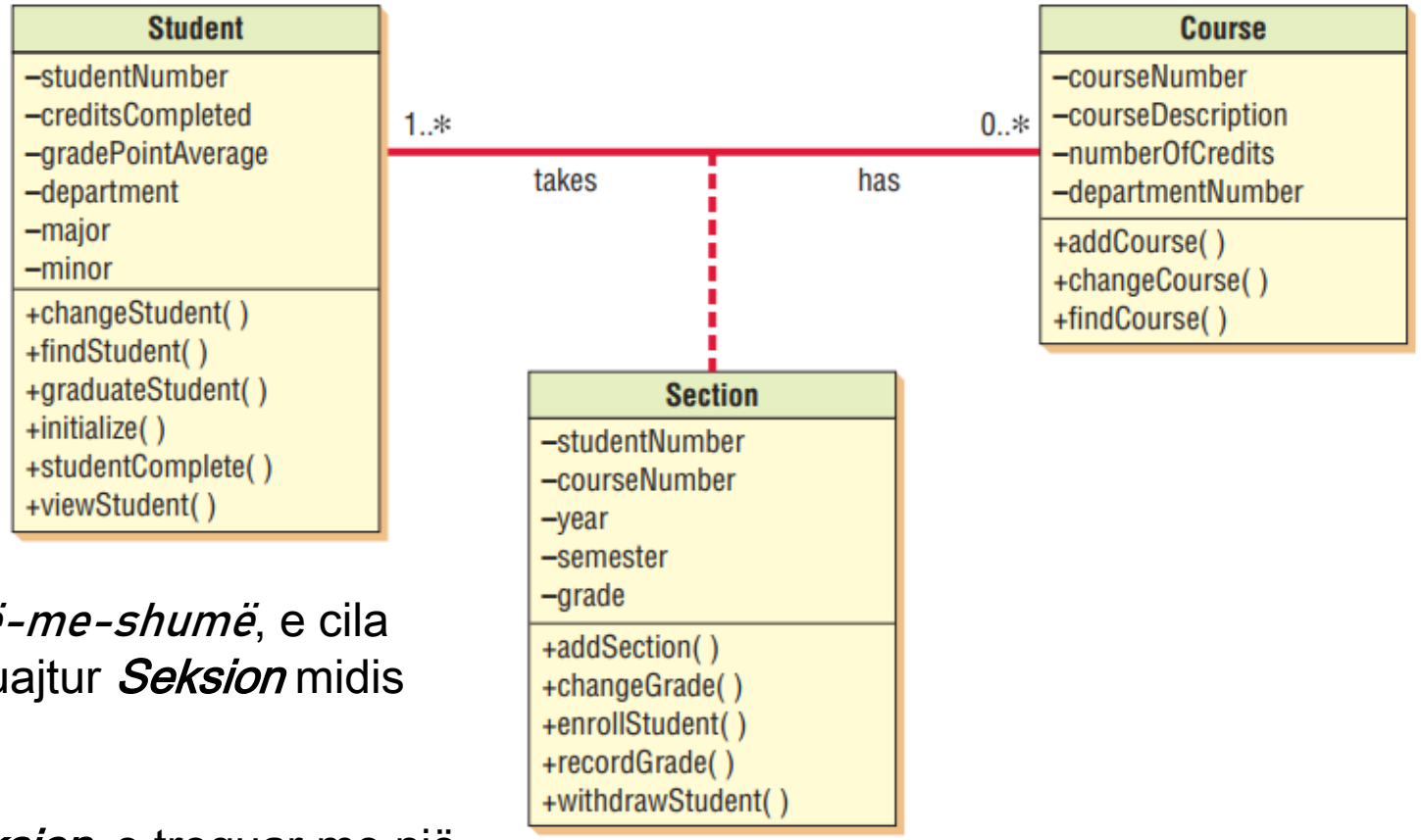

```java
public class Student
{
	//attributes
	private int studentID;
	private String firstName;
	private String lastName;
	private String street;
	private String city;
	private String state;
	private String zipCode;
	private Date dateAdmitted;
	private float numberCredits;
	private String lastActiveSemester;
	private float lastActiveSemesterGPA;
	private float gradePointAverage;
	private String major;

	//constructors
	public Student (String inFirstName, String inLastName, String inStreet,
		String inCity, String inState, String inZip, Date inDate)
	{
		firstName = inFirstName;
		lastName = inLastName;
		...
	}
	public Student (int inStudentID)
	{
		//read database to get values
	}

	//get and set methods
	public String getFullName ( )
	{
		return firstName + " " + lastName;
	}
	public void setFirstName (String inFirstName)
	{
		firstName = inFirstName;
	}
	public float getGPA ( )
	{
		return gradePointAverage;
	}
	//and so on

	//processing methods
	public void updateGPA ( )
	{
		//access course records and update lastActiveSemester and
		//to-date credits and GPA
	}
}
```

llojet e ***asociacionev*** që mund të kemi në ***diagramet e Klasës***





Shembull: një ***klase asociative*** në të cilën një ***seksion*** i veçantë përcakton relacionet midis
Studentit dhe **një Kursi**



Studenti dhe **Kursi** kanë një relacion *shumë-me-shumë*, e cila zgjidhet duke shtuar një klasë asocuar të quajtur ***Seksion*** midis klasave ***Studenti*** dhe ***Kursi***.

Diagrami ilustron një relacion të quajtur ***Seksion***, e treguar me një *linjë të ndërprer* të lidhur të *shumë-me-shumë* linja relacionesh

Një *Agregacion* përshkruhet shpesh si një relacion "ka një (has a)". Agregacioni ofron një mjet për të treguar se i gjithë objekti është i përbërë nga shuma e pjesëve të tij (objekte të tjera).
Në shembullin e ***regjistrimit të studentëve***, ***departamenti*** ka një ***kurs*** dhe ***kursi*** është për një ***departament***.
Ky relacion është një relacion më e dobët, sepse një ***departament*** mund të ndryshohet ose hiqet dhe ***kursi*** *mund të ekzistojë* ende. Diamanti në fund të linjës së relacioni nuk është plotë (jo i mbushur).

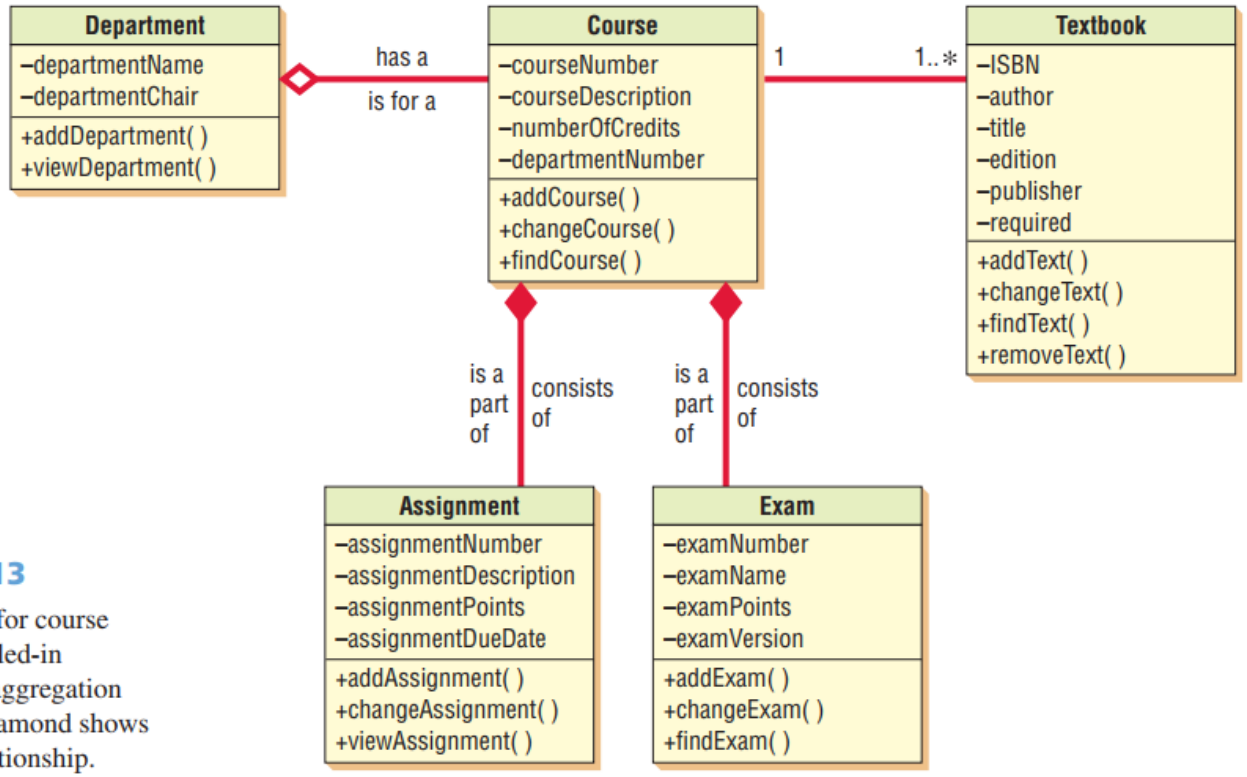
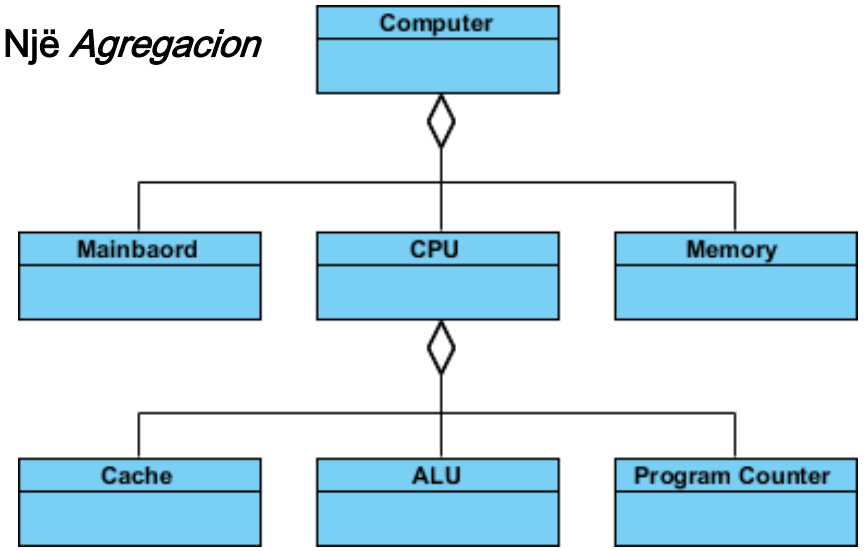


FIGURE 10.13
A class diagram for course offerings. The filled-in diamonds show aggregation and the empty diamond shows a whole-part relationship.

Një *Agregacion*



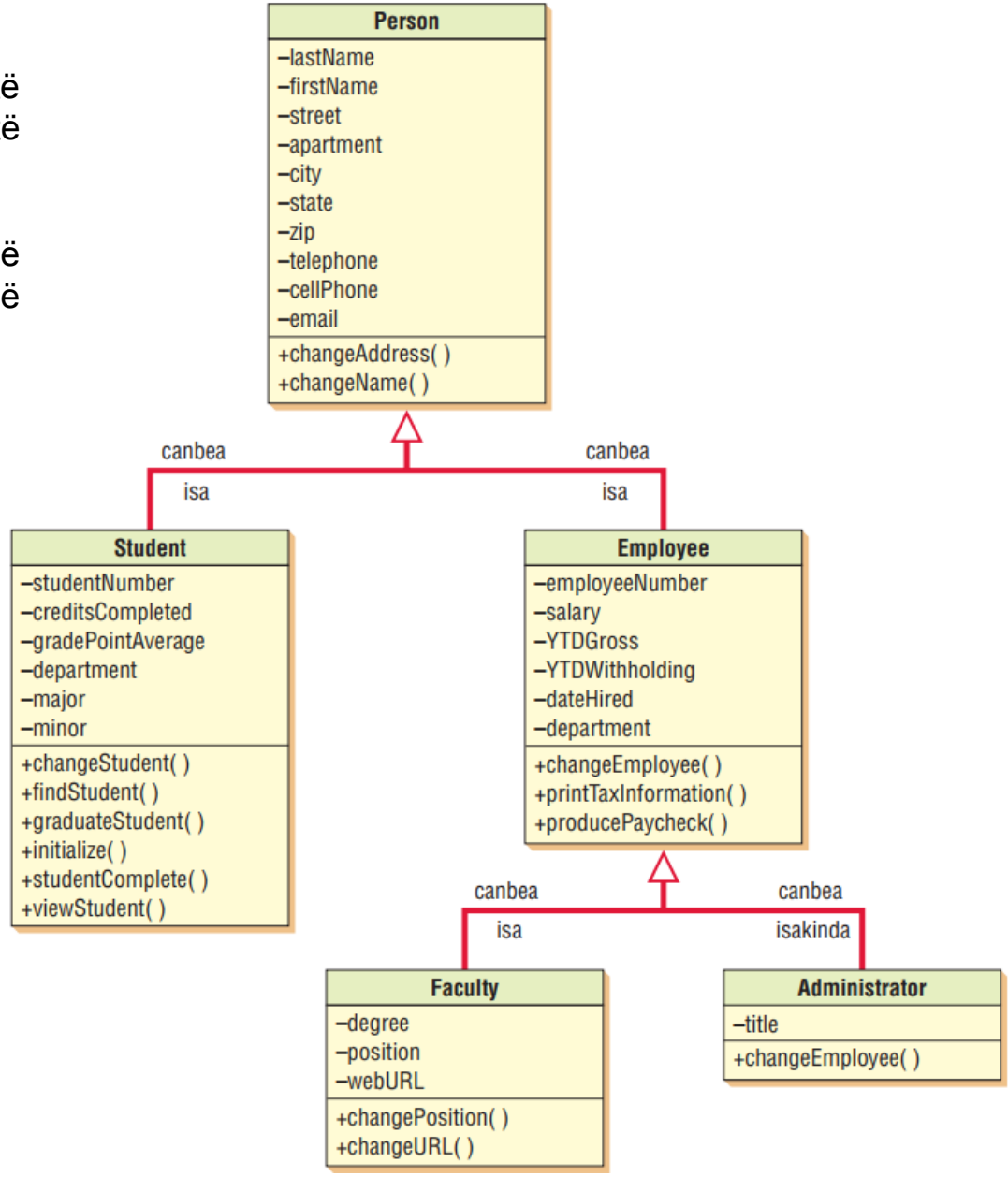
Kompozimi (Përbërja), një relacion *tërësa/pjesës* në të cilën e tërë ka një përgjegjësi për pjesën, është një relacion më e fortë dhe zakonisht tregohet me ***një diamant të mbushur***.
Nëse i tërë është fshirë, të gjitha pjesët fshihen. Për shembull, ekziston një relacion midis ***asigmenti*** (detyre) dhe një ***kursi***, si dhe midis një ***kursi*** dhe një ***provimi***. Nëse ***kursi*** është *fshirë*, ***asigmenti*** dhe ***provimi*** *fshihen* gjithashtu.

Një *gjeneralizim* përshkruan një relacion midis një lloji të *përgjithshëm* të gjërave dhe llojeve më *specifike* të gjërave. Ky lloj relacioneve është përshkruar shpesh si një relacion "***është një (is a)".***

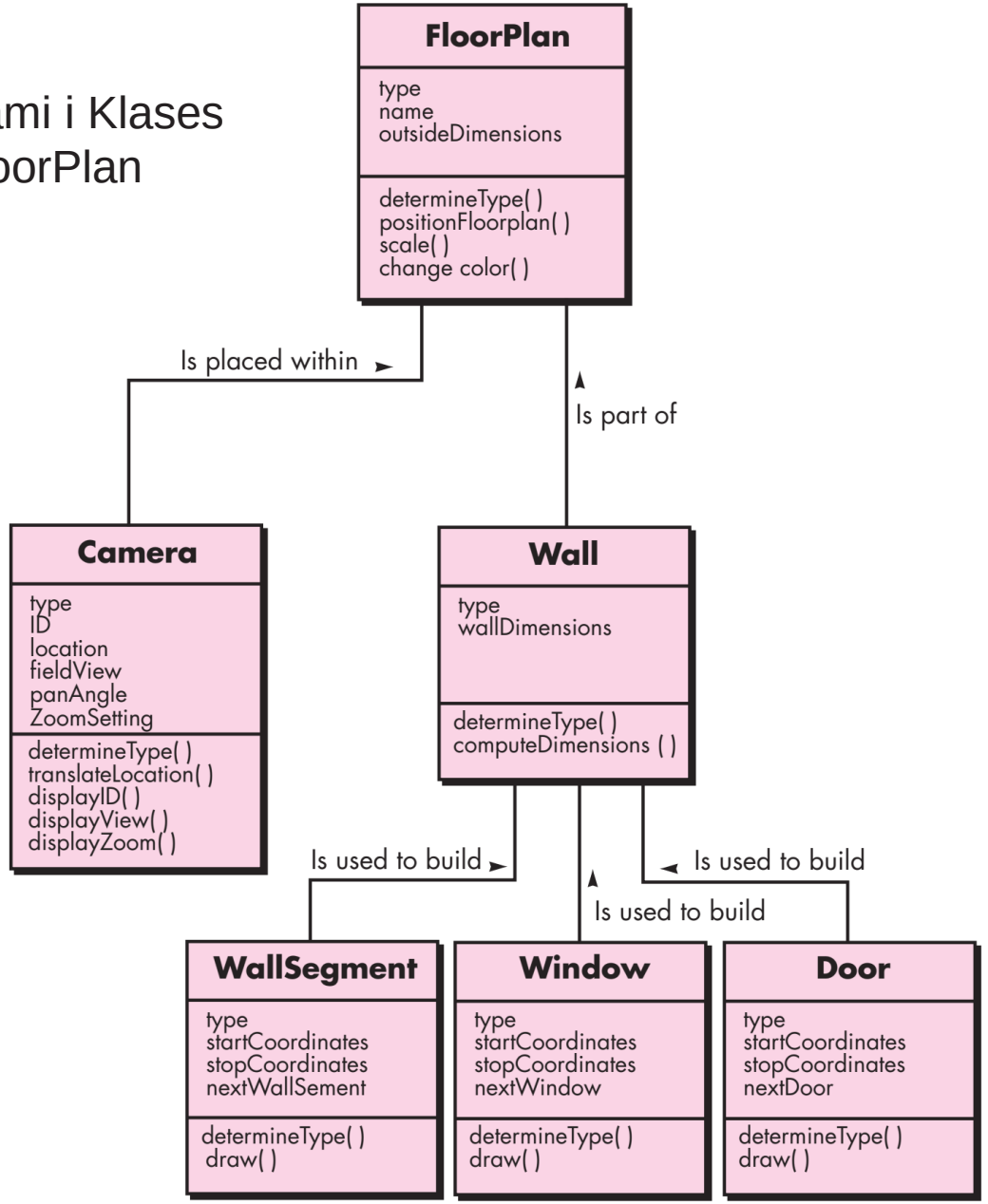
P.sh, një *makinë* është një *automjet* dhe një *kamion* është një *automjet*, në këtë rast, *automjeti* është një gjë gjenerale, ndërsa *makina* dhe *kamioni* janë gjërat më **specifike**.

Trashëgimia. Disa klasa mund të kenë të njëjtat *atribute* dhe/ose *metoda*. Kur kjo ndodhë, krijohet një *klasë e generale* që përmban *atributet* dhe *metodat* e përbashkta. Klasa e specializuar trashëgon ose merr *atributet* dhe *metodat* e klasës së gjenerale.

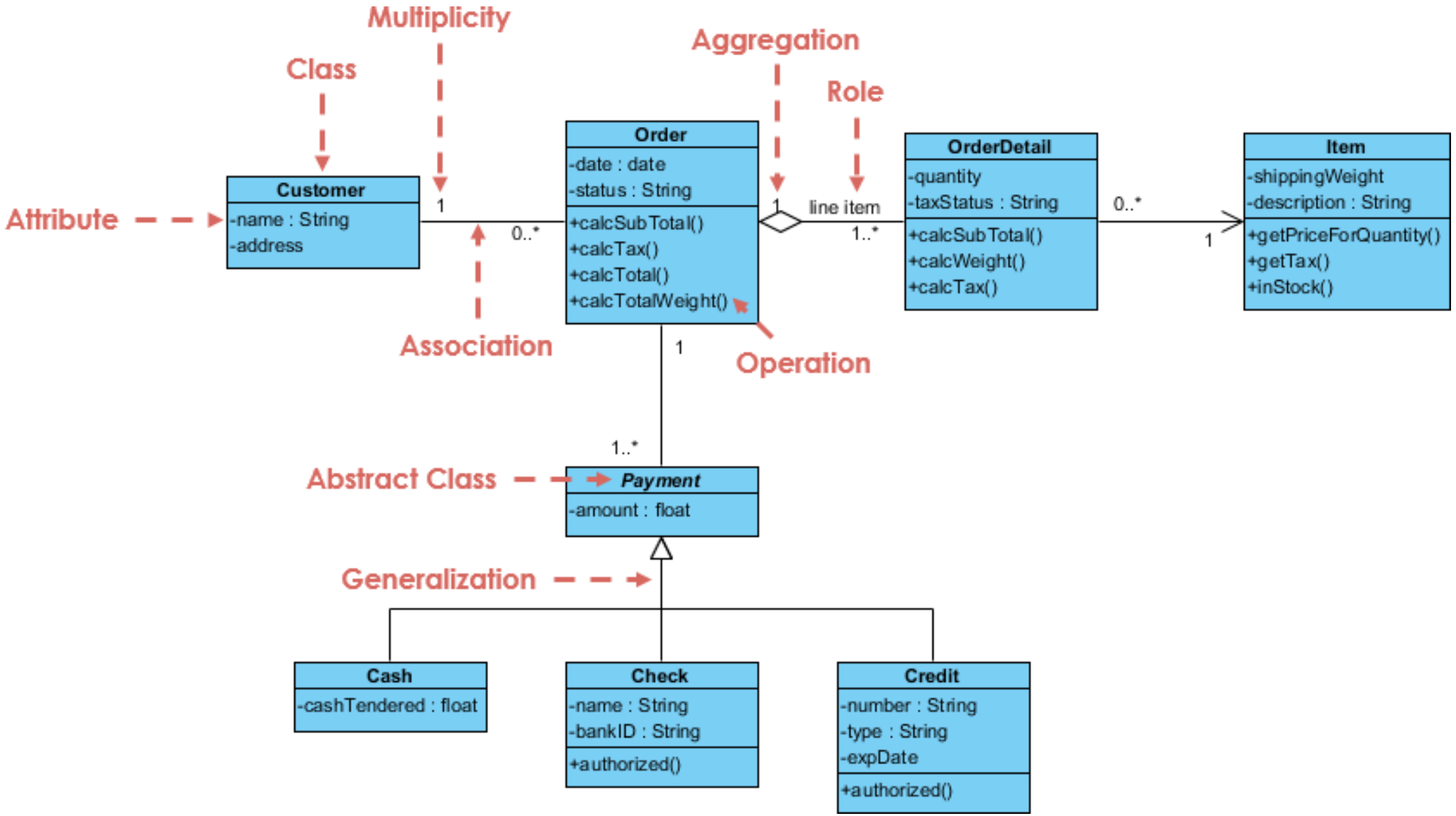
Përveç kësaj, klasa e *specializuar* ka *atribute* dhe *metoda* që janë **unike** dhe të përcaktuara ose definuar ***vetëm*** në *klasën e specializuar*



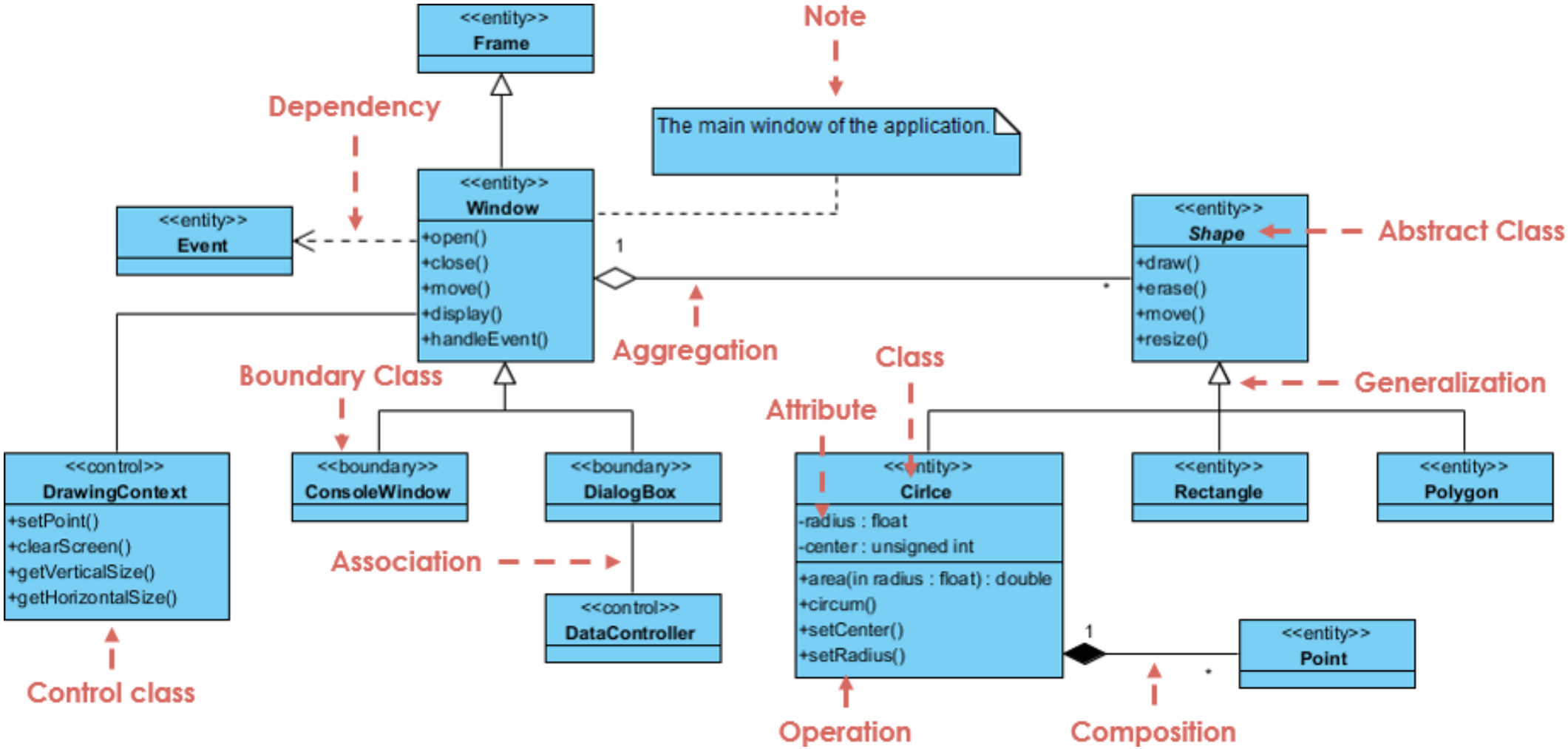
Diagrami i Klases për FloorPlan

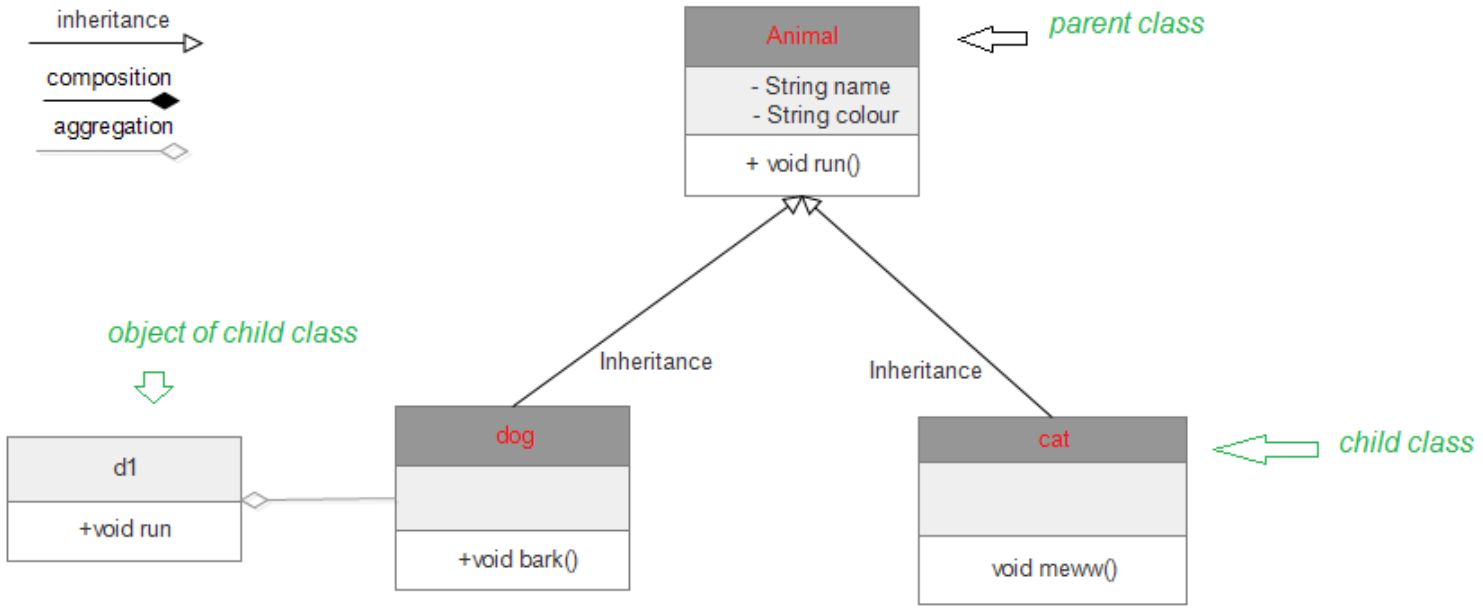


Diagrami i klasës në UML, Rasti i blerjës online



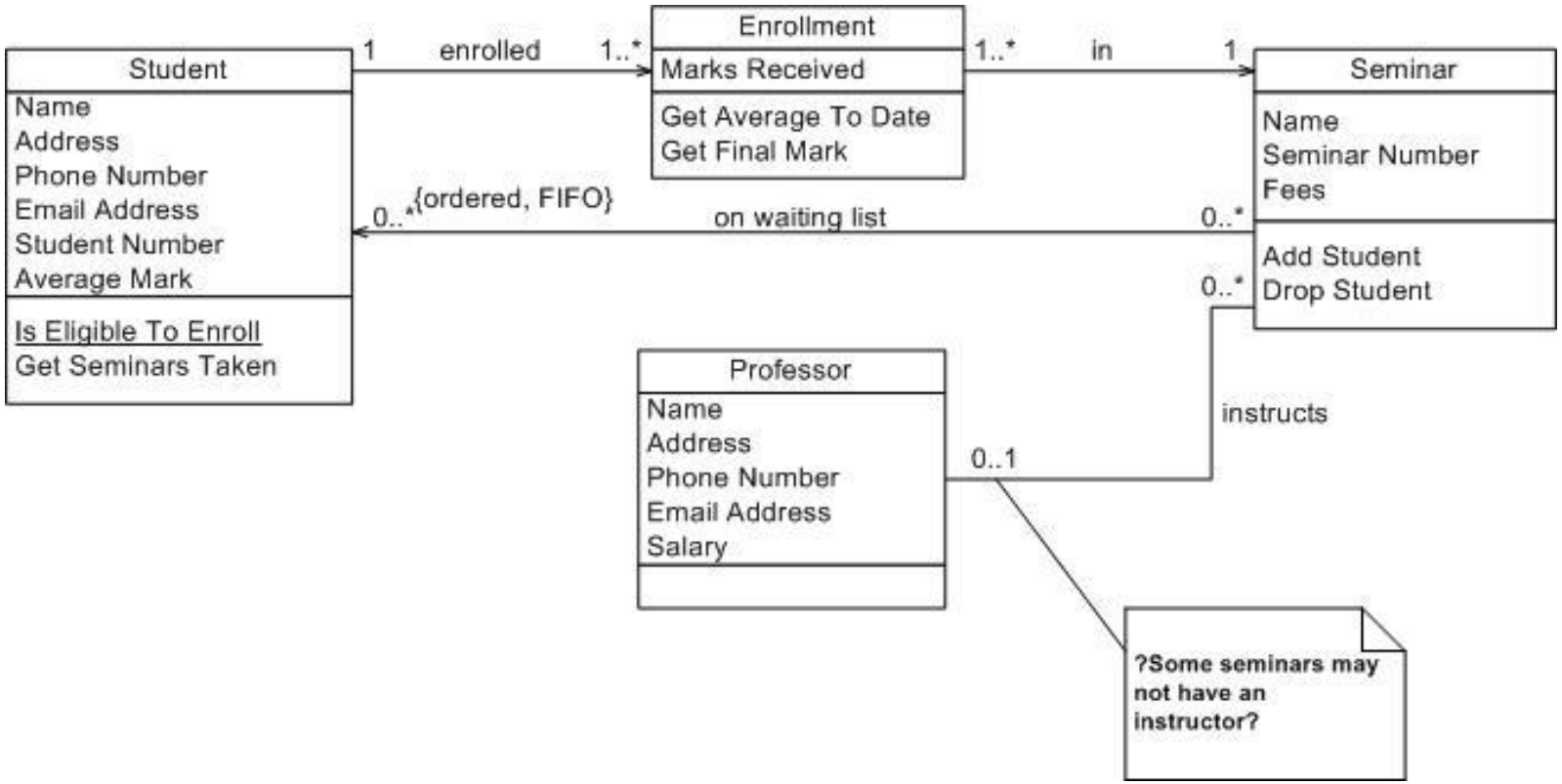
Diagrami i klasës në UML, Rasti i GUI-it






```java
class GFG {
    public static void main(String[] args)
    {
        dog d1 = new dog();
        d1.bark();
        d1.run();
        cat c1 = new cat();
        c1.meww();
    }
}

class Animal {
    public void run()
    {
        String name;
        String colour;

        System.out.println("animal is running");
    }
}

class dog extends Animal {
    public void bark()
    {
        System.out.println("wooh!wooh! dog is barking");
    }
    public void run()
    {
        System.out.println("dog is running");
    }
}

class cat extends Animal {
    public void meww()
    {
        System.out.println("meww! meww!");
    }
}
```

Diagrami i klasës në UML.

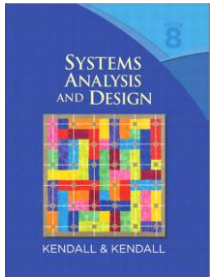


Kur përdoret diagramet e klasës

- Sa herë që doni të modelit/dizajnoni për gjërat në një sistem, dhe se si ato lidhen me njëri-tjetrin (i quajtur nganjëherë ***modelimit të dhënave***).
- Jep një pamje të përqëndruar në të dhëna të dobishme për:
  - Planifikimi i klasave që nevojiten në OOP
  - Vendimarrje mbi skemën për bazat e të dhënave
- Por keni kujdes që të mos anashkaloni në detaje!
  - E zakonshme për të injoruar disa aspekte
  - P.sh. Modelimi e Klasve dhe asociacionet, por jo veti (atributet) ose operacione

Referencat

- **Kapitulli 10**: Systems analysis and design 8 Ed. By Kenneth E. Kendall



- **Kapitulli 5**: Software Engineering. 9th ed. By Ian Sommerville