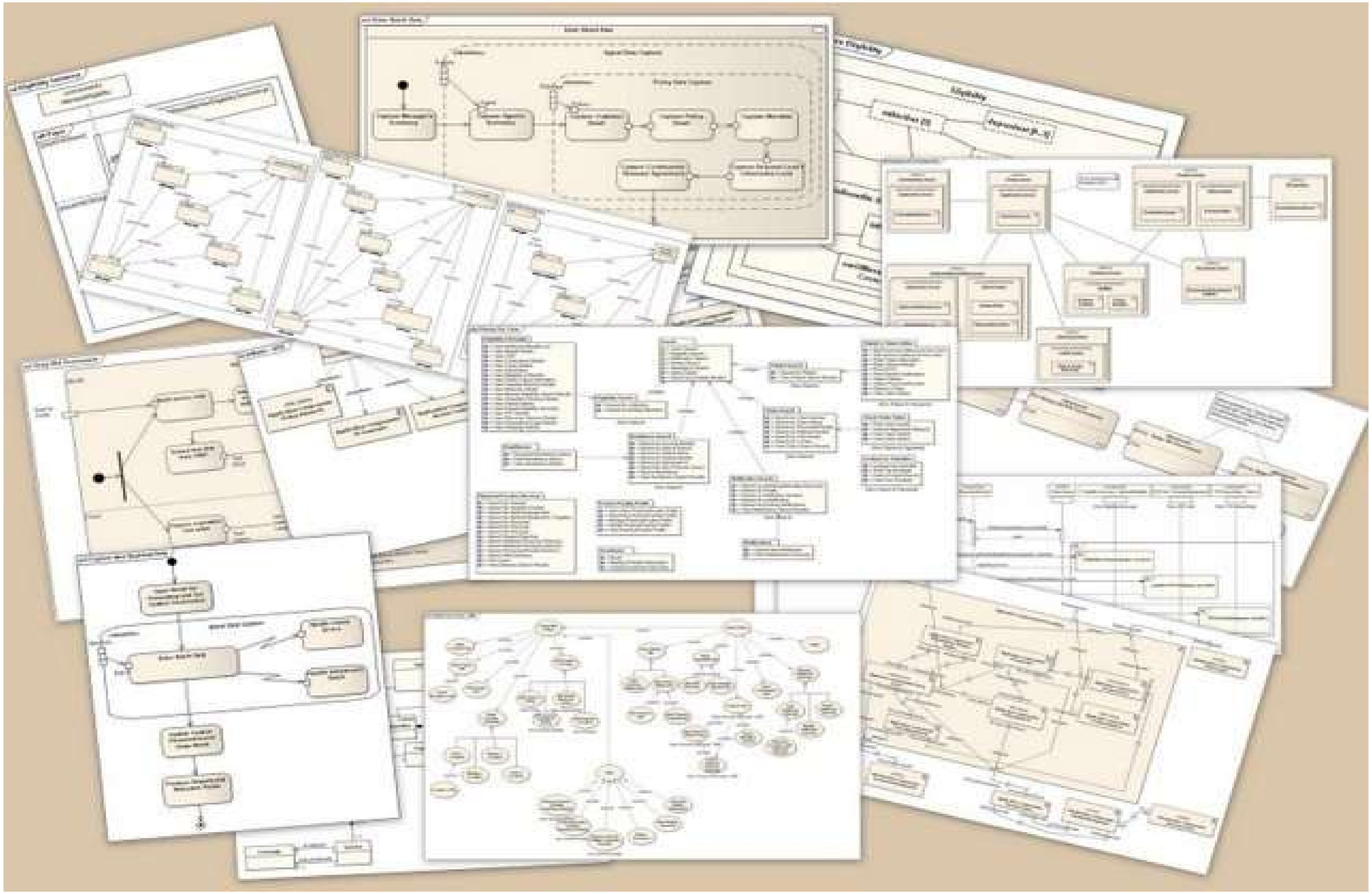
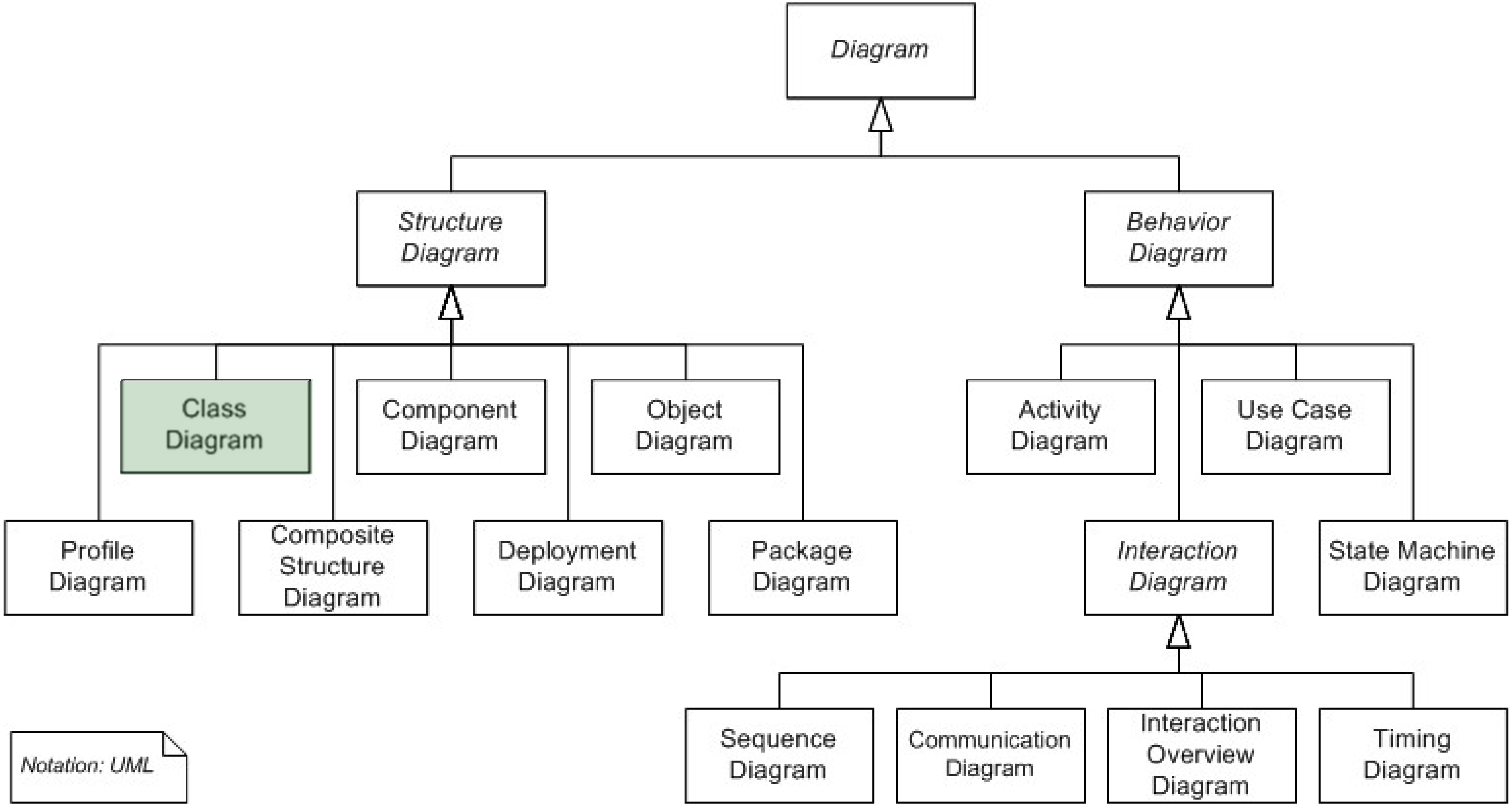
D12 UNIT 6-B CONFIGURATION MANAGEMENT AND PROTOTYPE

Domain model of the prototype

UML
Unified Modeling Language

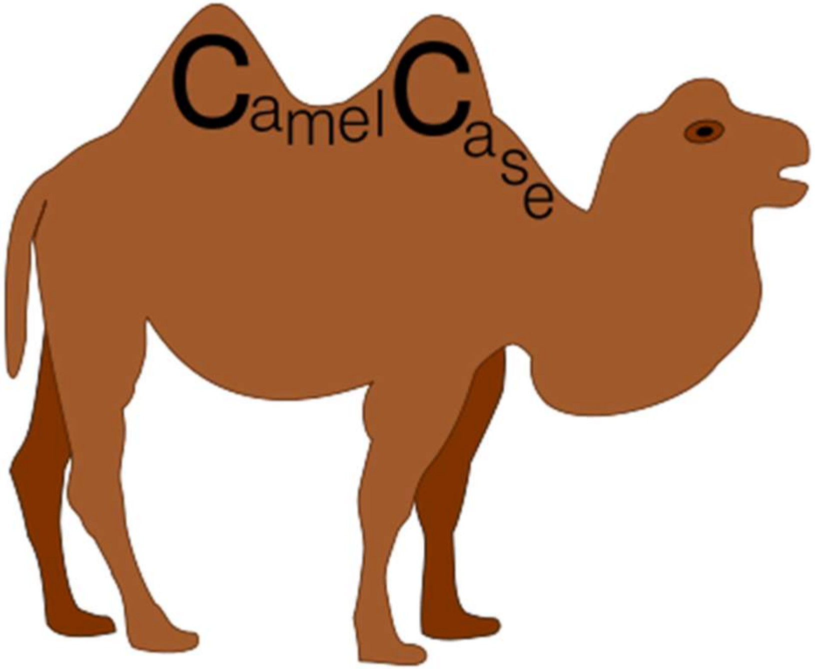


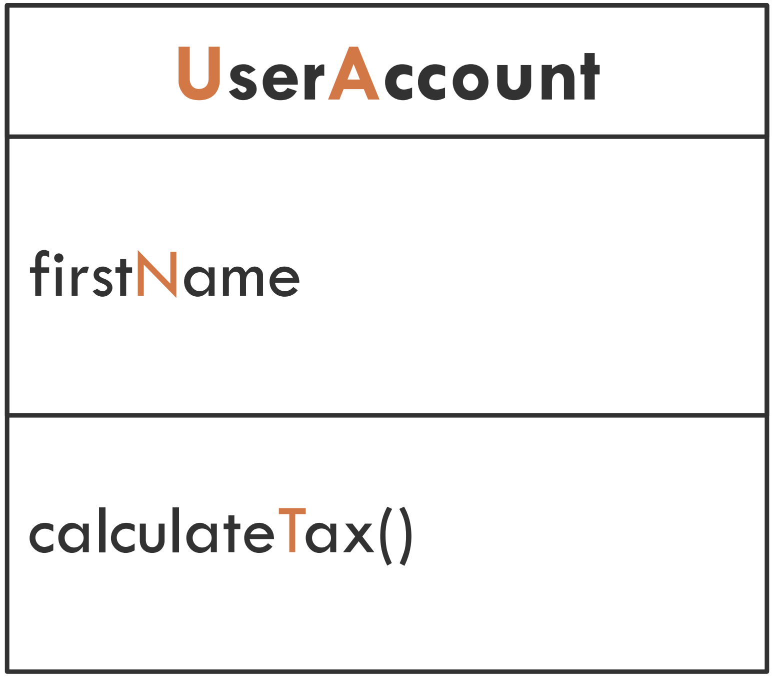
UML
Unified Modeling Language



Classes

CamelCase





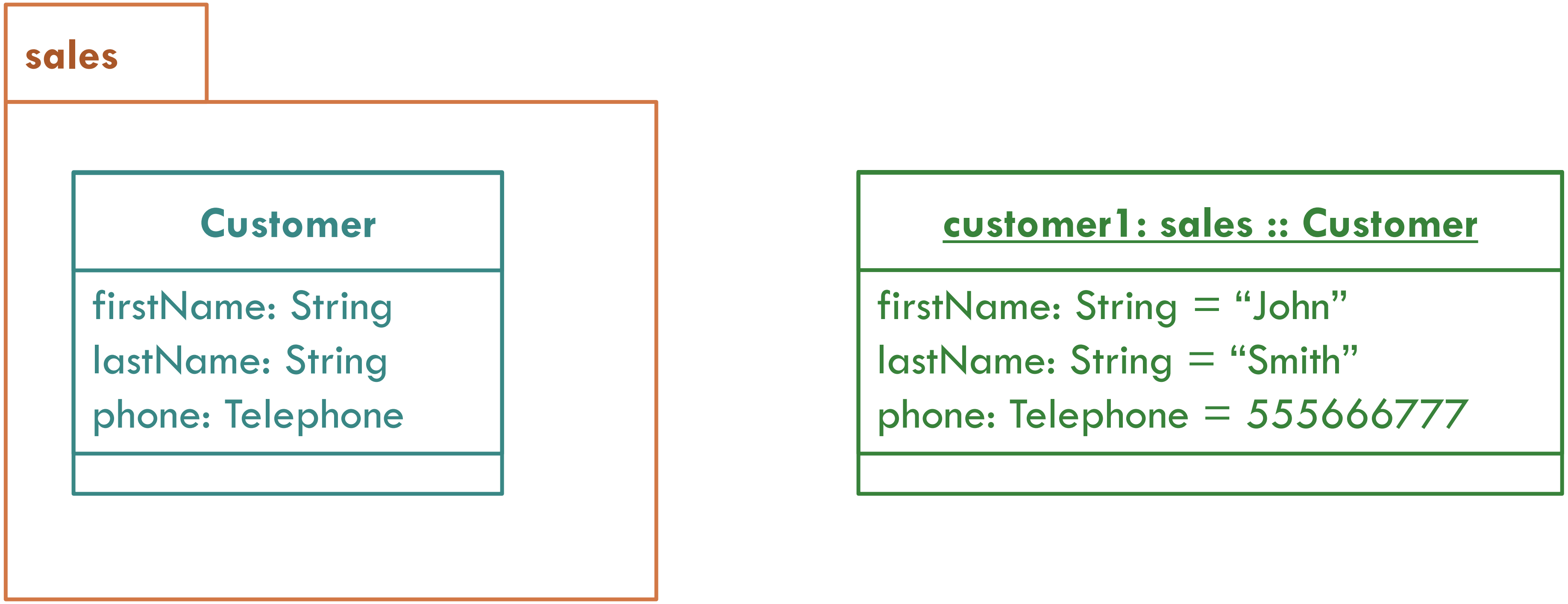
UserAccount

firstName

calculateTax()

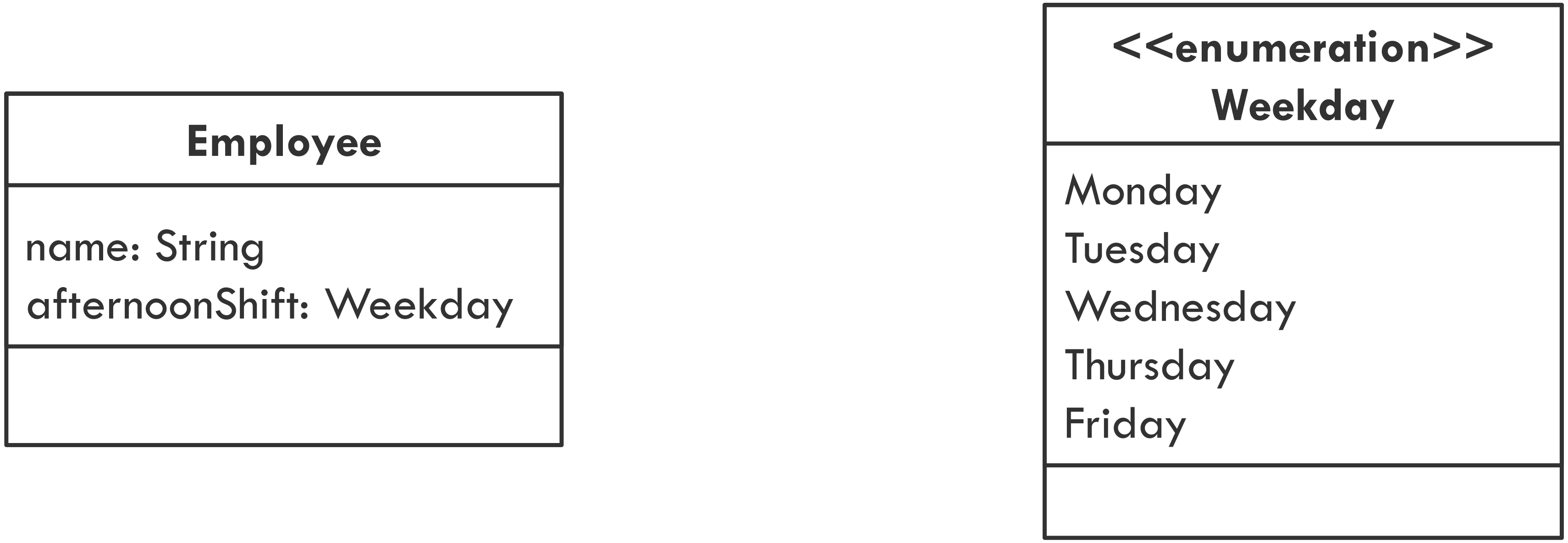
Classes

Package, class and instance


sales
Customer
firstName: String
lastName: String
phone: Telephone
customer1: sales :: Customer
firstName: String = “John”
lastName: String = “Smith”
phone: Telephone = 555666777

Attributes
Enumeration

Visibility Name Multiplicity : Type = Initial Value

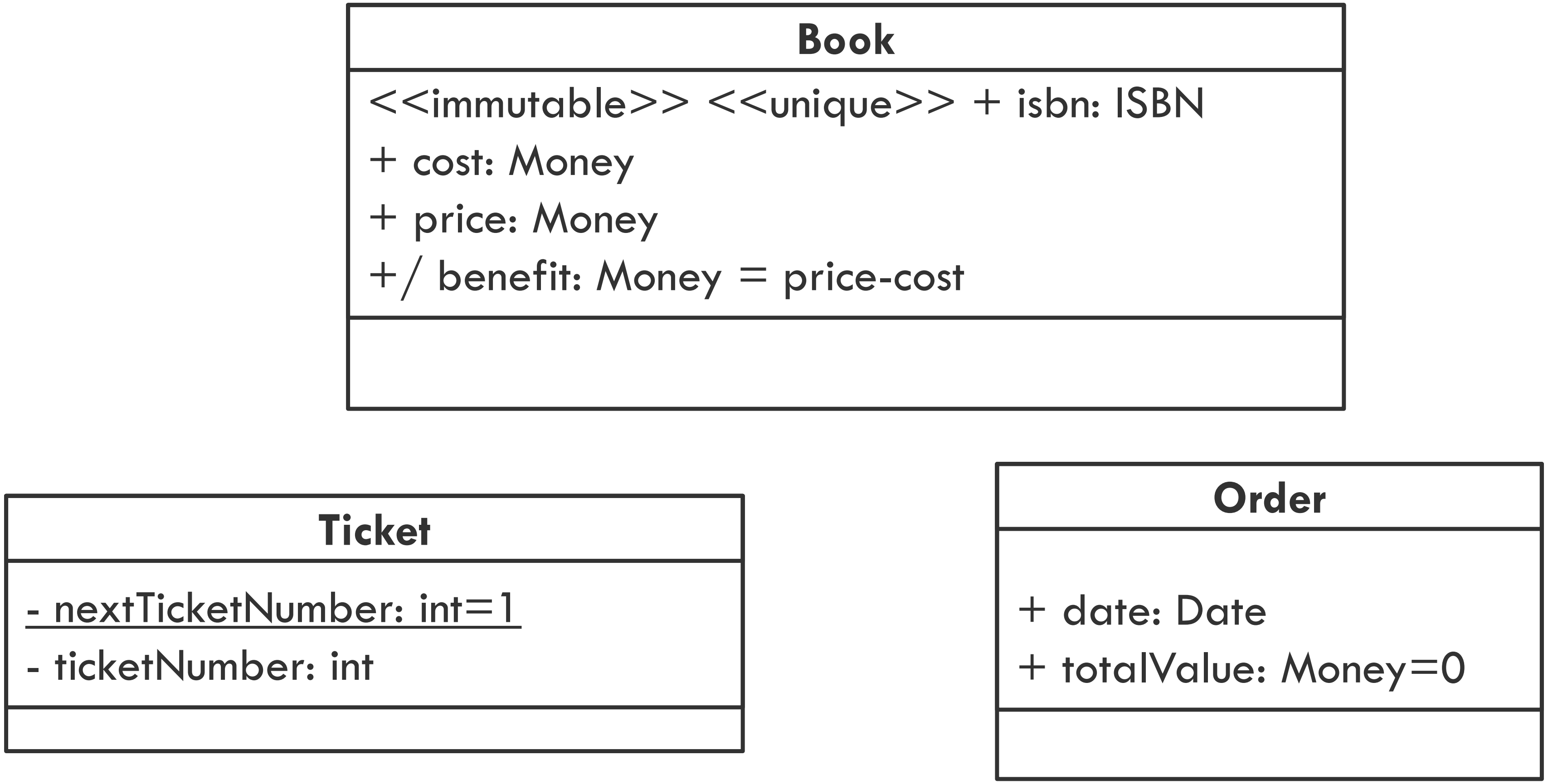


```
enum Weekday{
          Monday, Tuesday, Wednesday, Thursday, Friday
}
```

Attributes

Initial value, derived, unique, immutable and static

Visibility Name Multiplicity : Type = Initial Value



Book

<<immutable>> <<unique>> + isbn: ISBN

+ cost: Money

+ price: Money

+/ benefit: Money = price-cost

Ticket

- nextTicketNumber: int=1

- ticketNumber: int

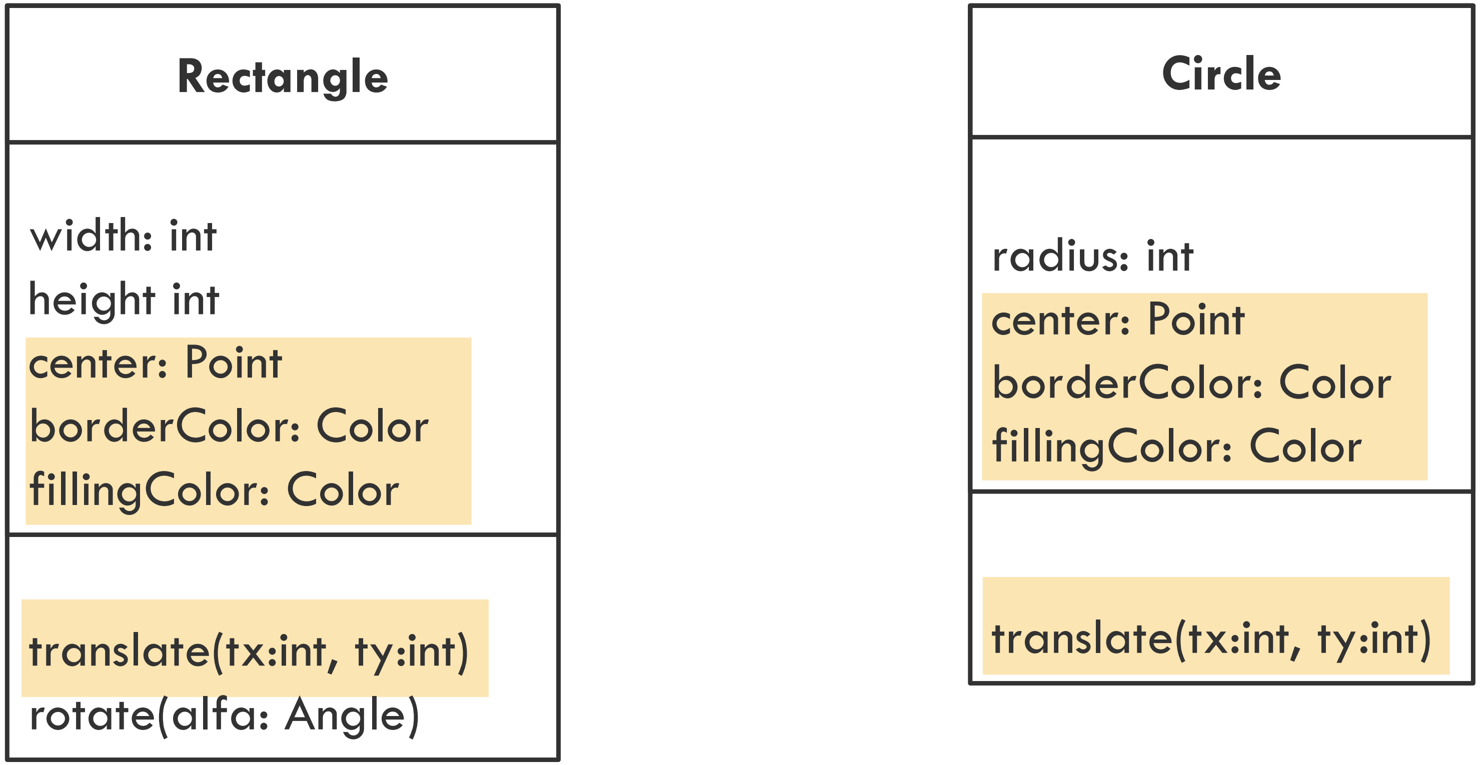
Order

+ date: Date

+ totalValue: Money=0

Generalization

Specialization, inheritance and abstract classes



Rectangle

width: int
height int
center: Point
borderColor: Color
fillingColor: Color

translate(tx:int, ty:int)
rotate(alfa: Angle)

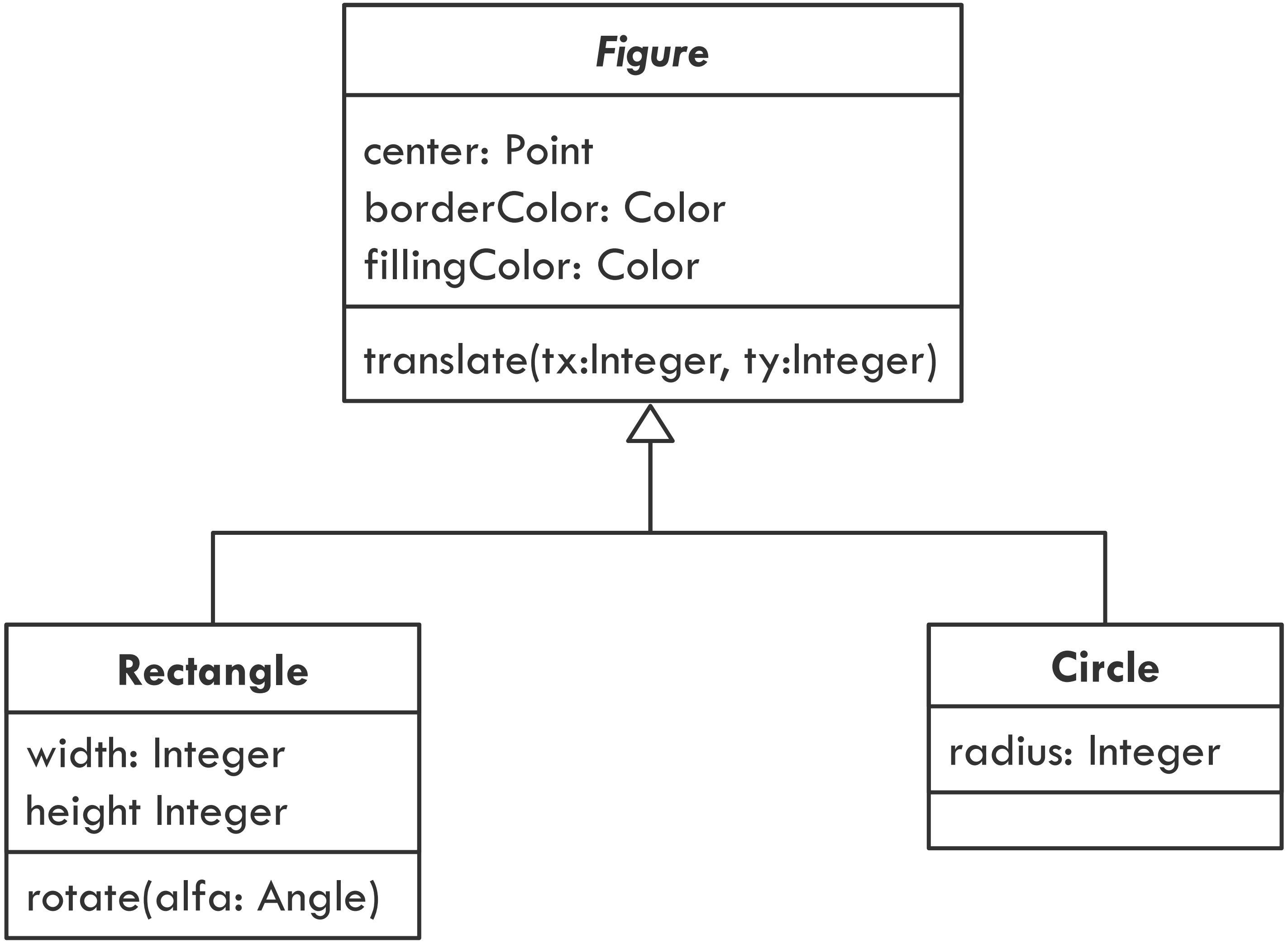
Circle

radius: int
center: Point
borderColor: Color
fillingColor: Color

translate(tx:int, ty:int)

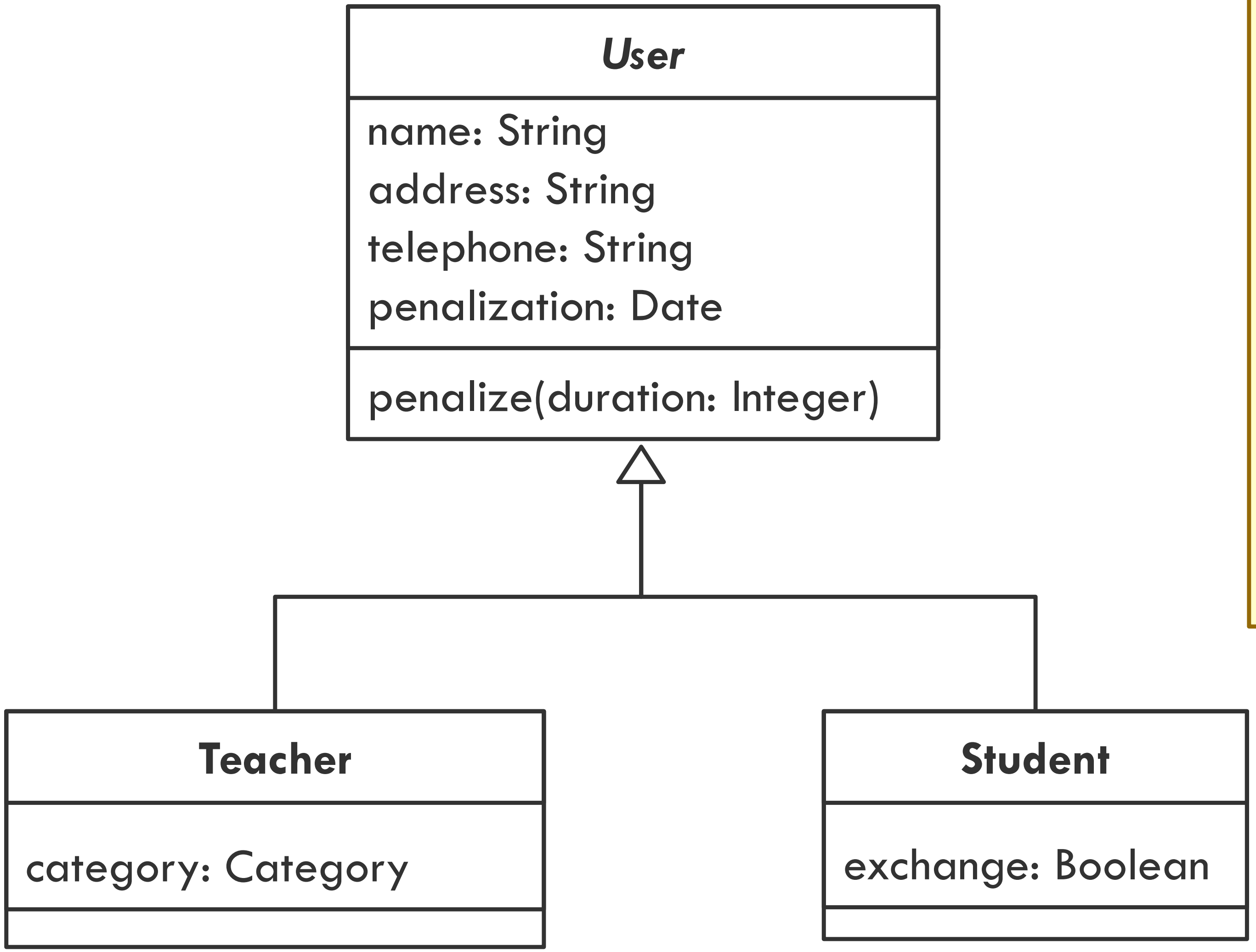
Generalization

Specialization, inheritance and abstract classes



Generalization

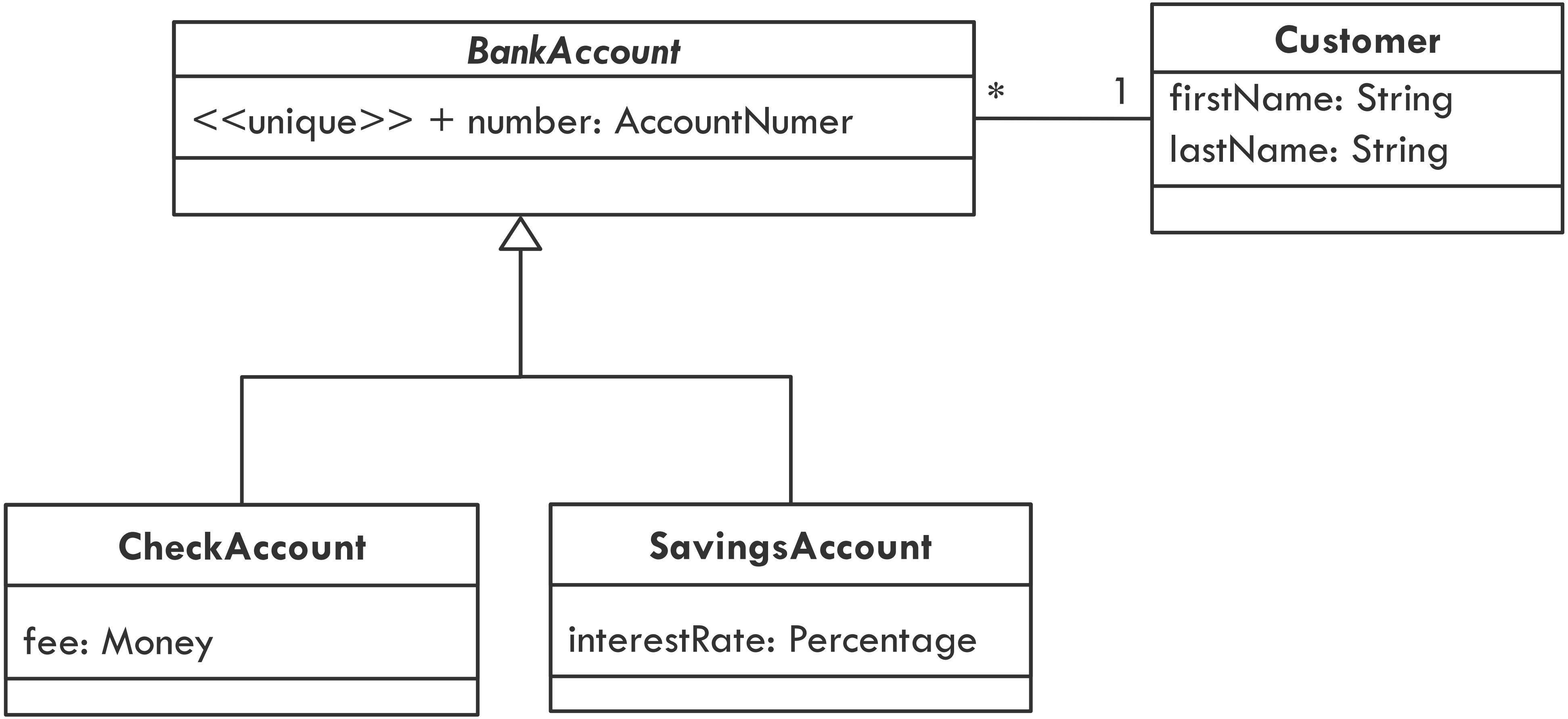
Specialization, inheritance and abstract classes



```
abstract class User{
        ...
}

class Teacher extends User{
        ...
}

class  Student extends User{
        ...
}
```

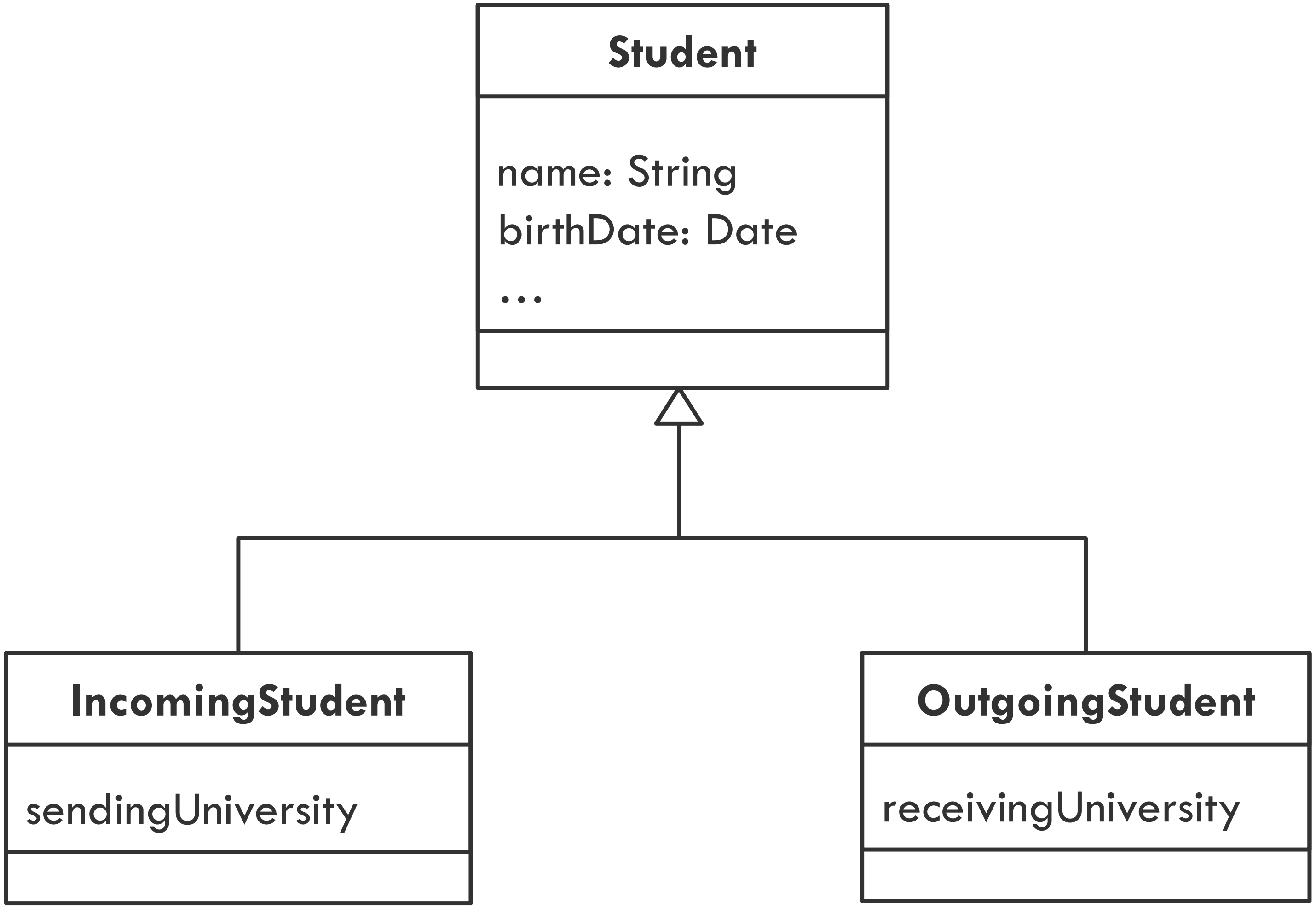
Generalization

Specialization, inheritance and abstract classes



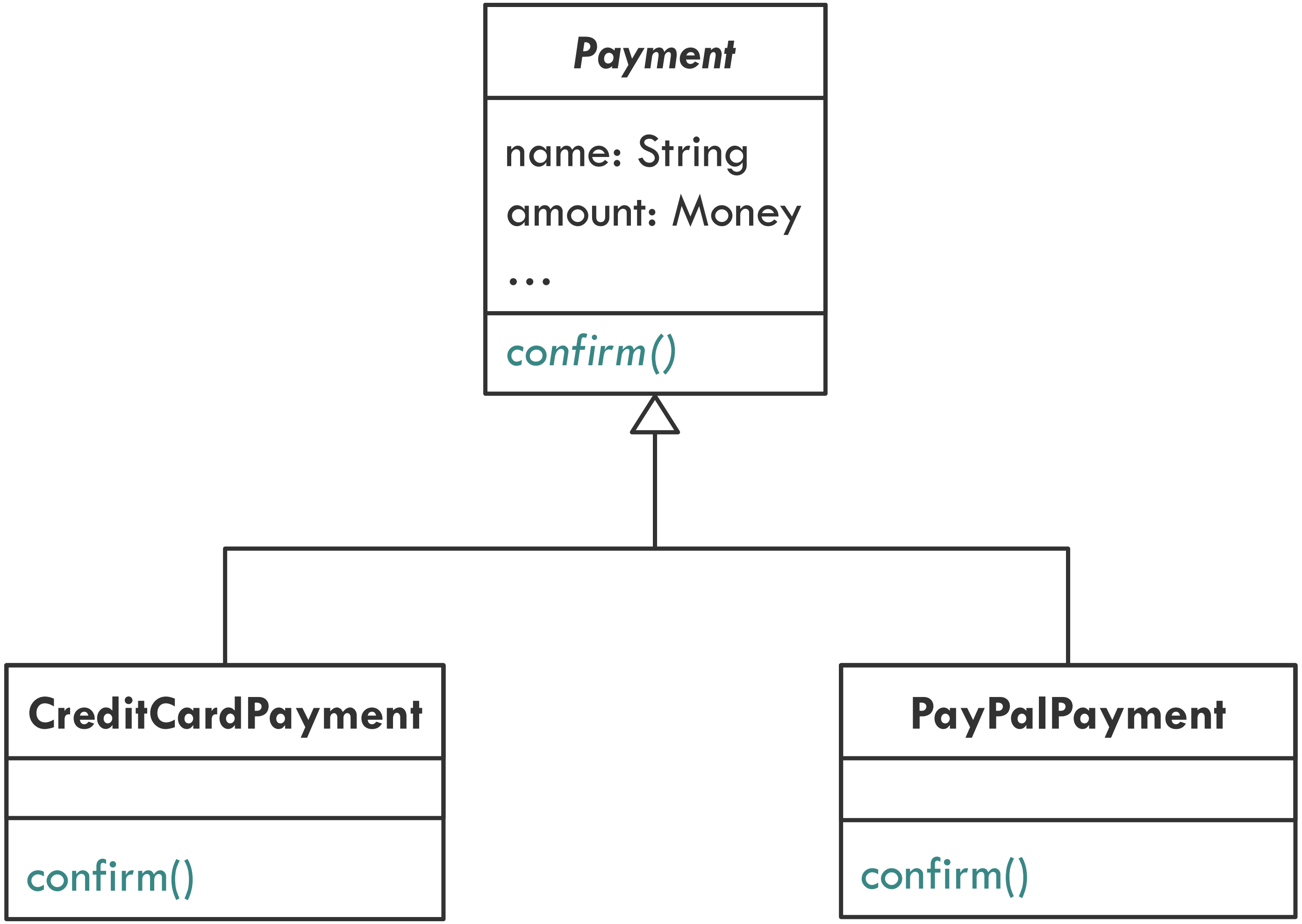
Generalization

Specialization, inheritance and abstract classes



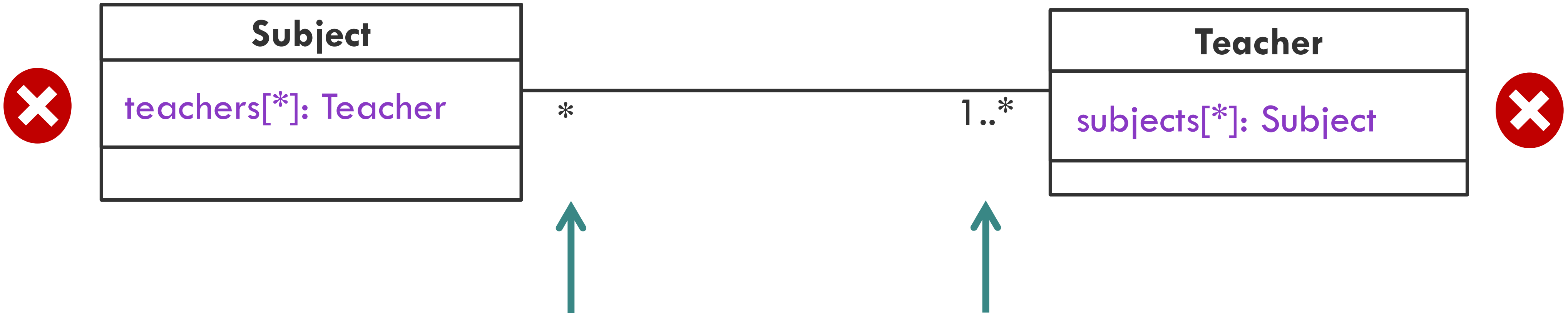
Generalization

Polymorphic operations



Association

Multiplicity/cardinality



How many subjects does a teacher teach?

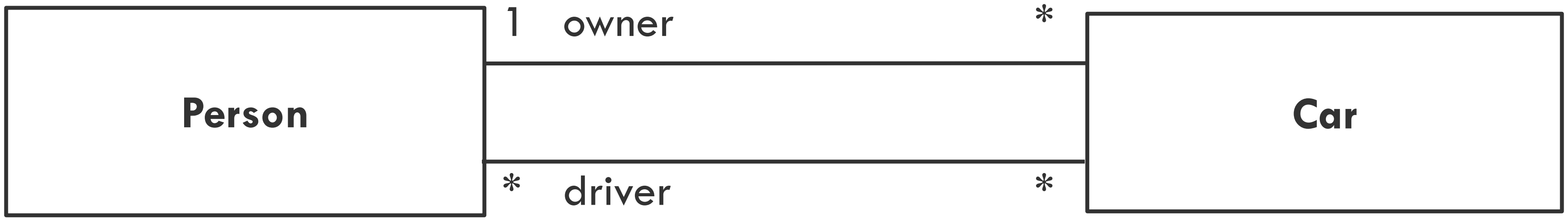
How many teachers teach a subject?

Implicit attributes and methods (not indicated)
Multiplicities are analyzed in a stable state.

```
class Subject{
        …
        ArrayList<Teacher> teachers;
        …
}
```

```
class Teacher{
        …
        ArrayList<Subject> subjects;
        …
}
```

Association
Roles



1

0..1

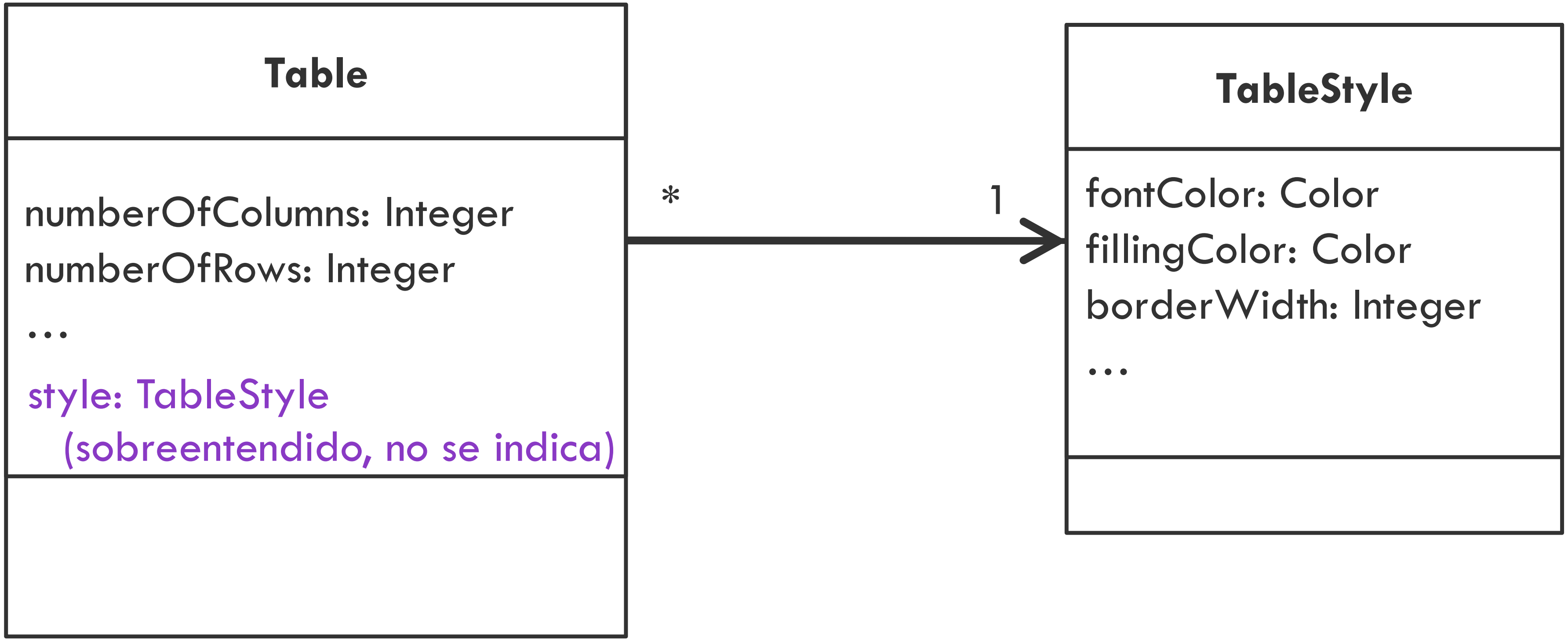
* (also 0..*)

1..*

, = o

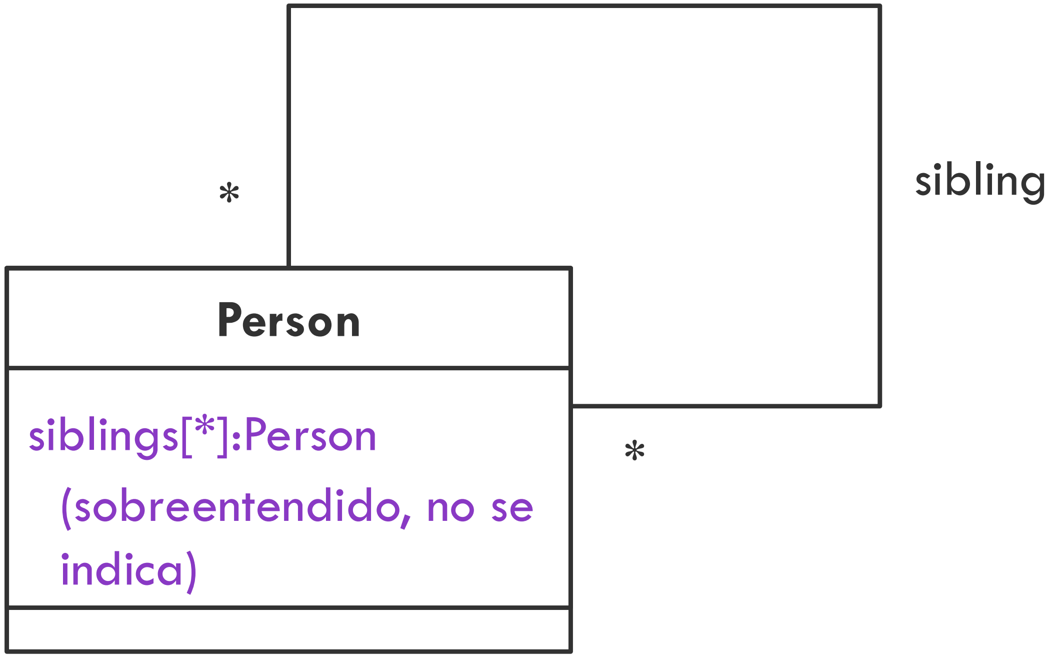
.. = from/up to

Association
Navigability



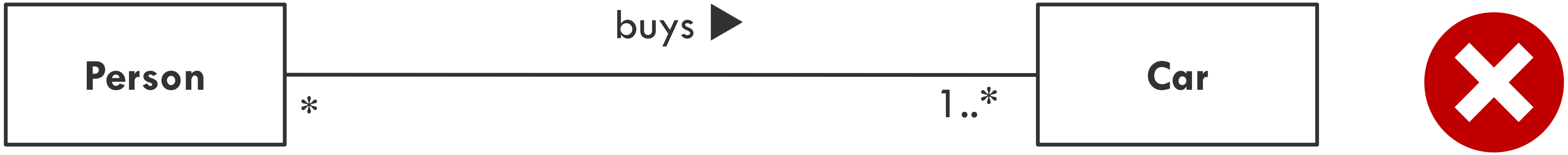
Association

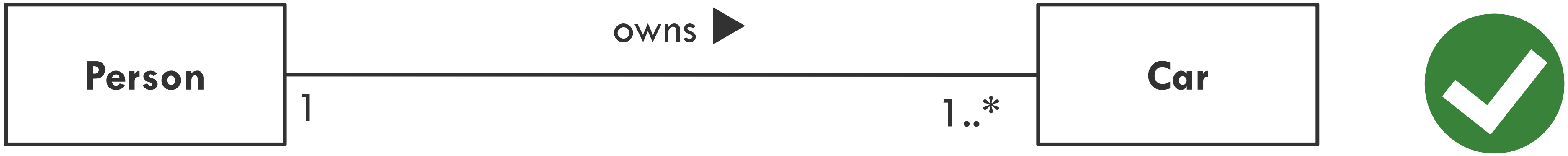
Self-association or reflexive association

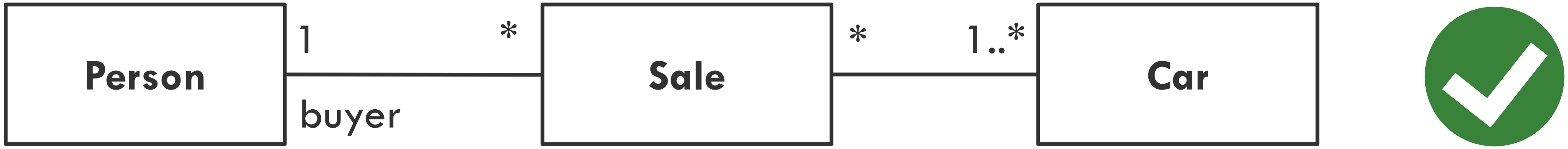


Association

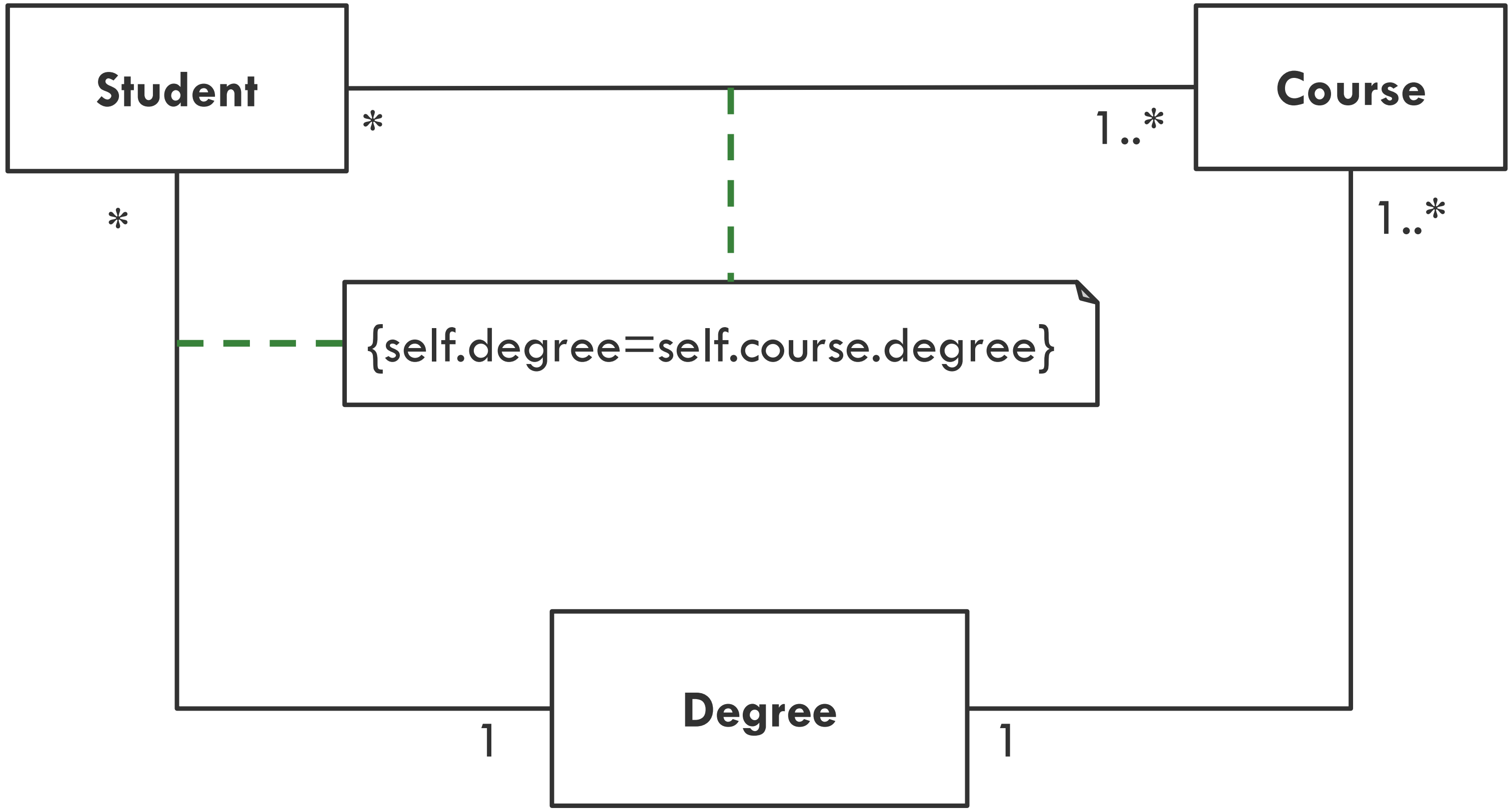
Associations vs operations





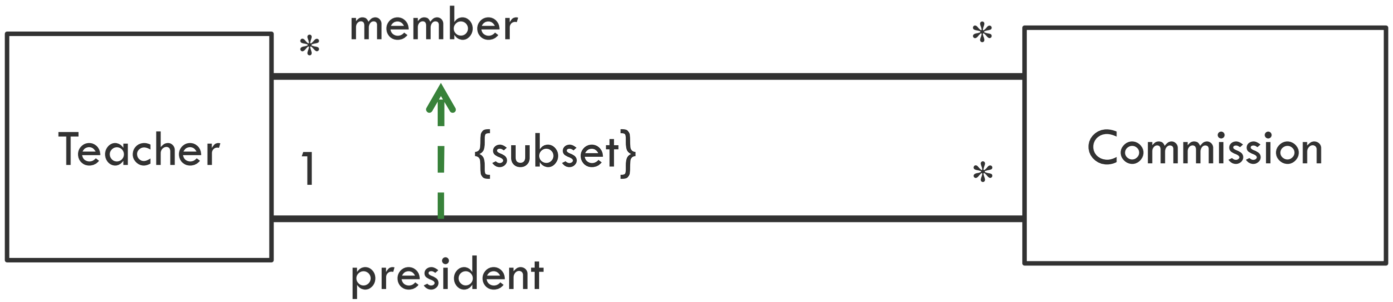


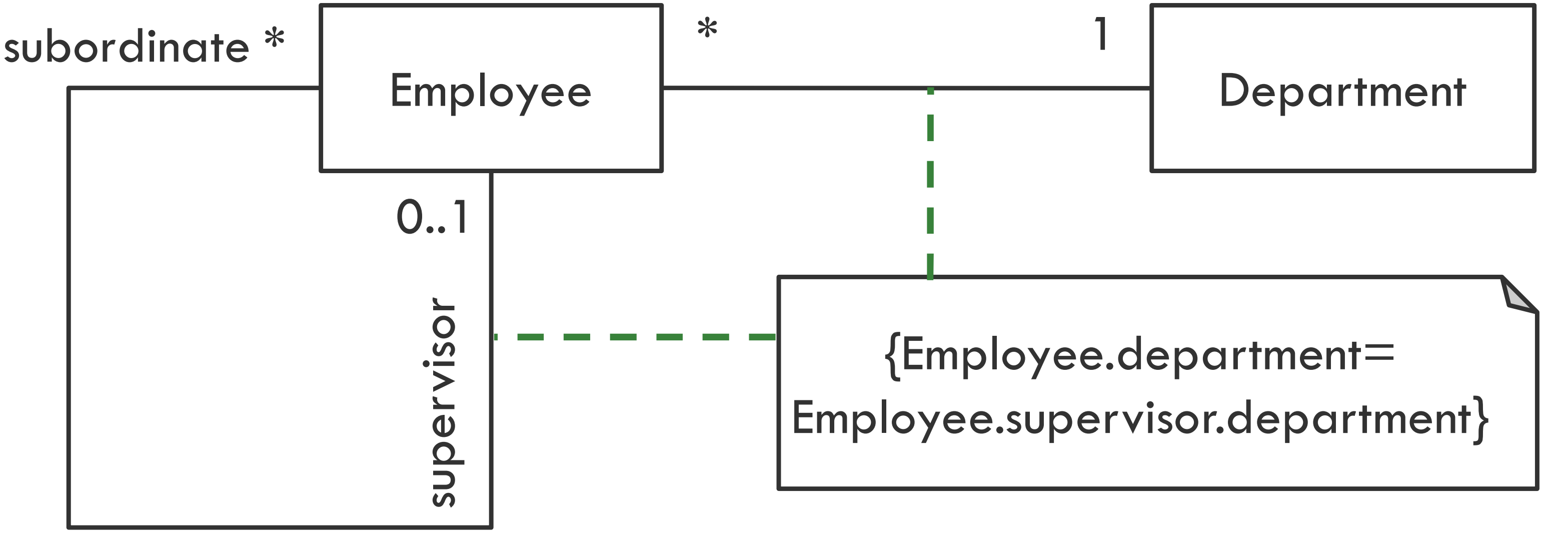
Association
Constraints



{self.degree=self.course.degree}

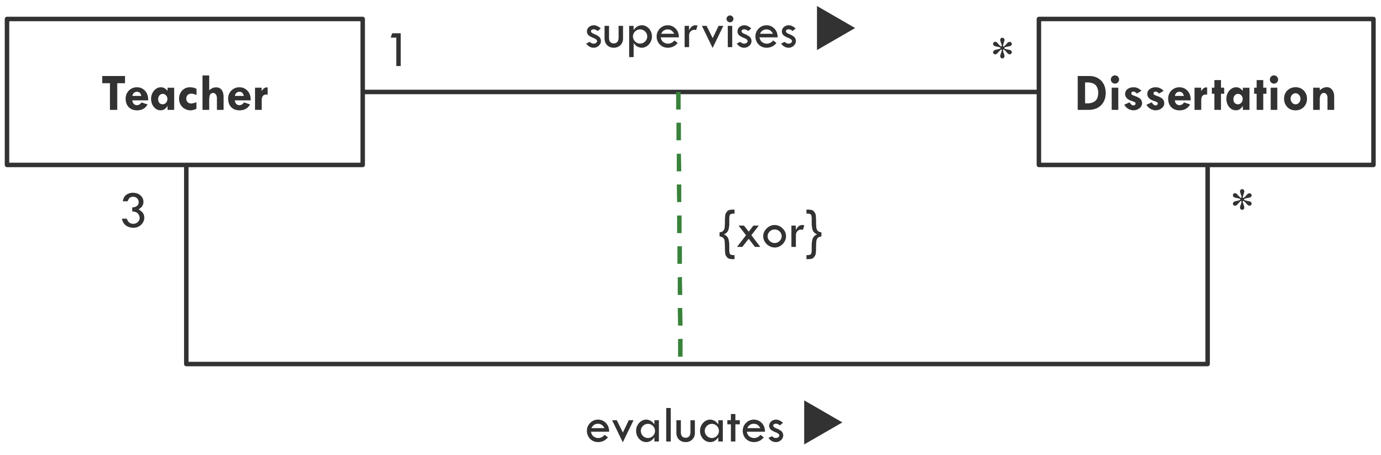
Association
Constraints





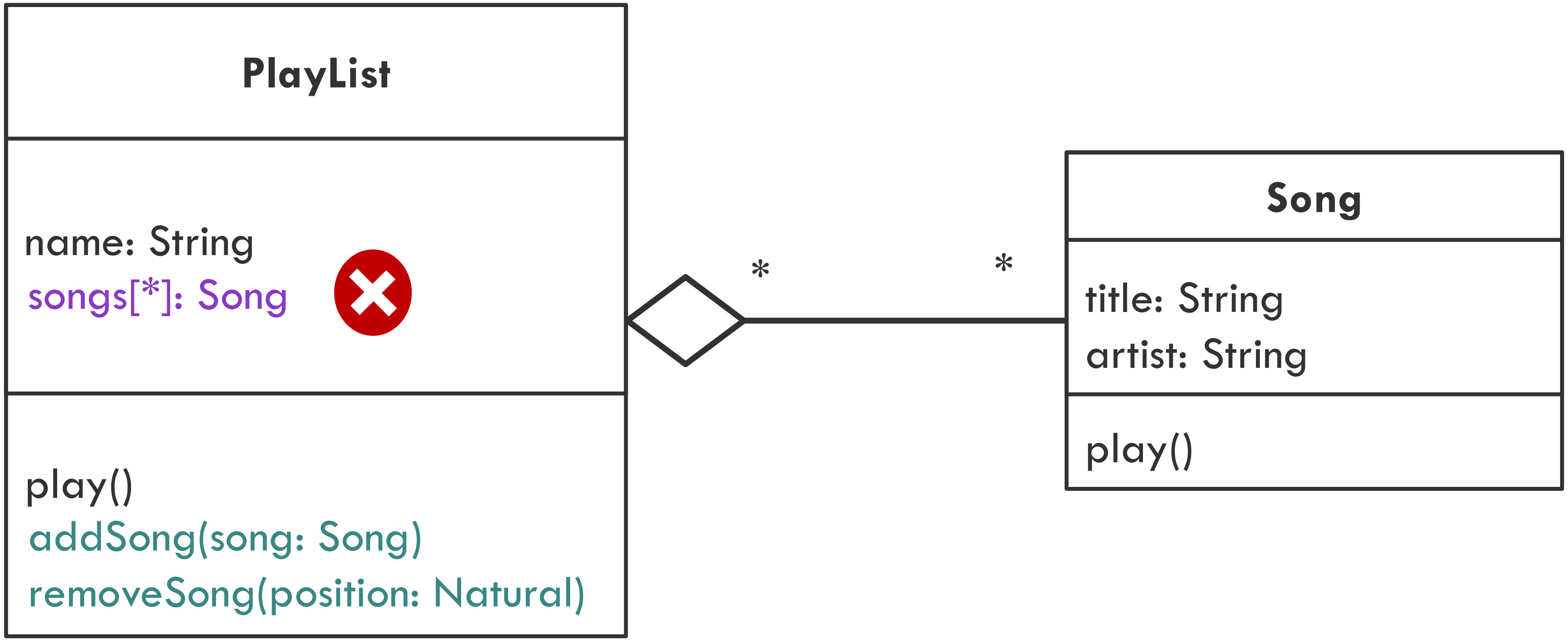
{Employee.department=
Employee.supervisor.department}

Association
Constraints



Restriction on the association, not on the multiplicity.

Aggregation
Shared Aggregation



PlayList

name: String
songs[*]: Song ✖

play()
addSong(song: Song)
removeSong(position: Natural)

* *

Song

title: String
artist: String

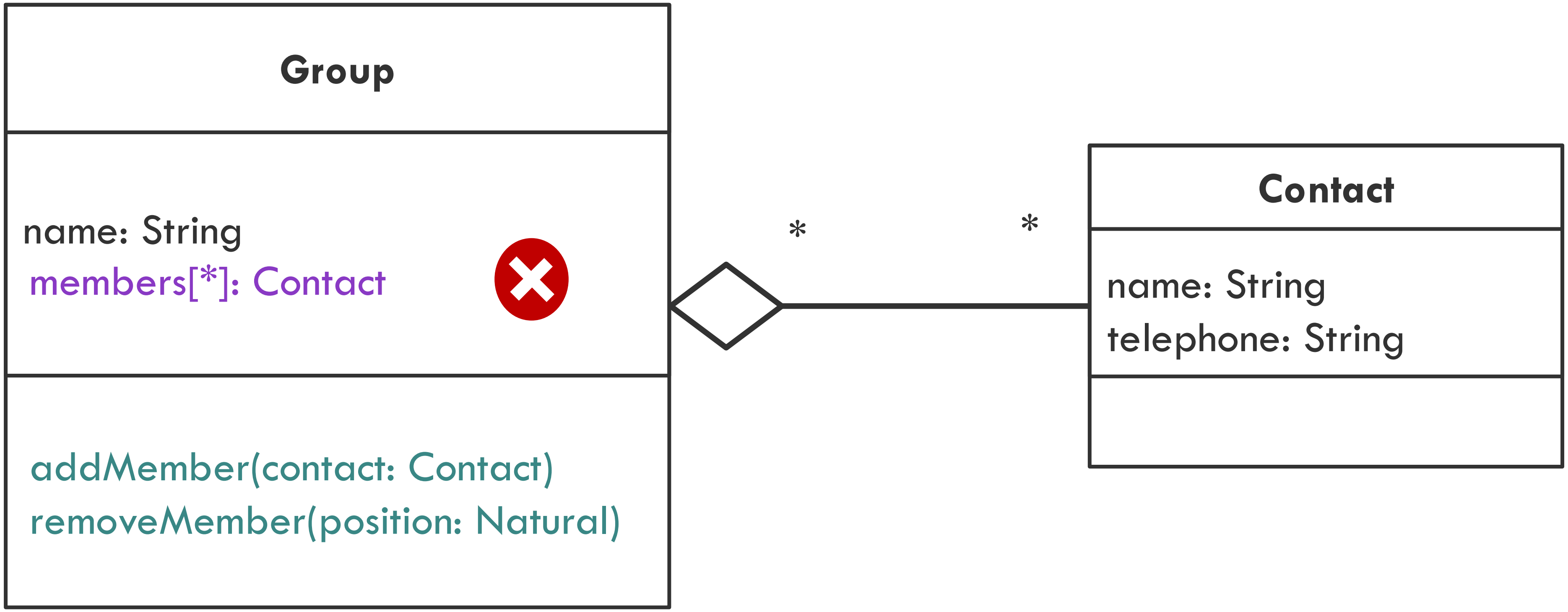
play()

Whole

Part

Aggregation

Shared Aggregation



Group

name: String
members[*]: Contact ✖

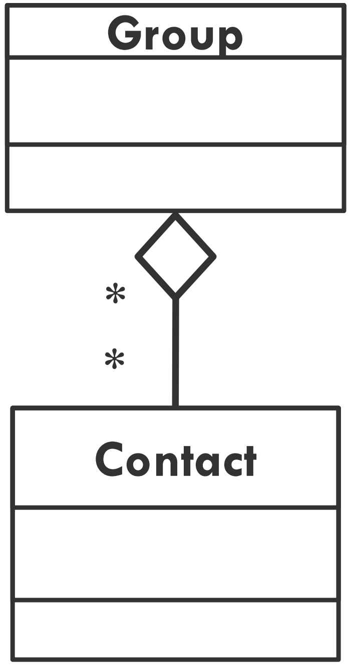
addMember(contact: Contact)
removeMember(position: Natural)

* *

Contact

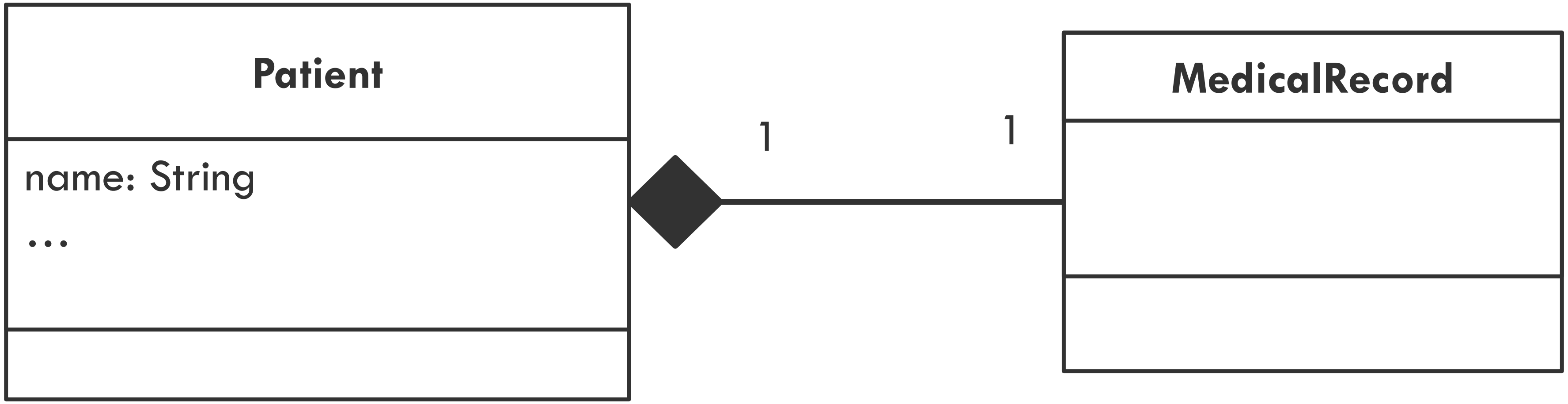
name: String
telephone: String

Aggregation
Shared Aggregation



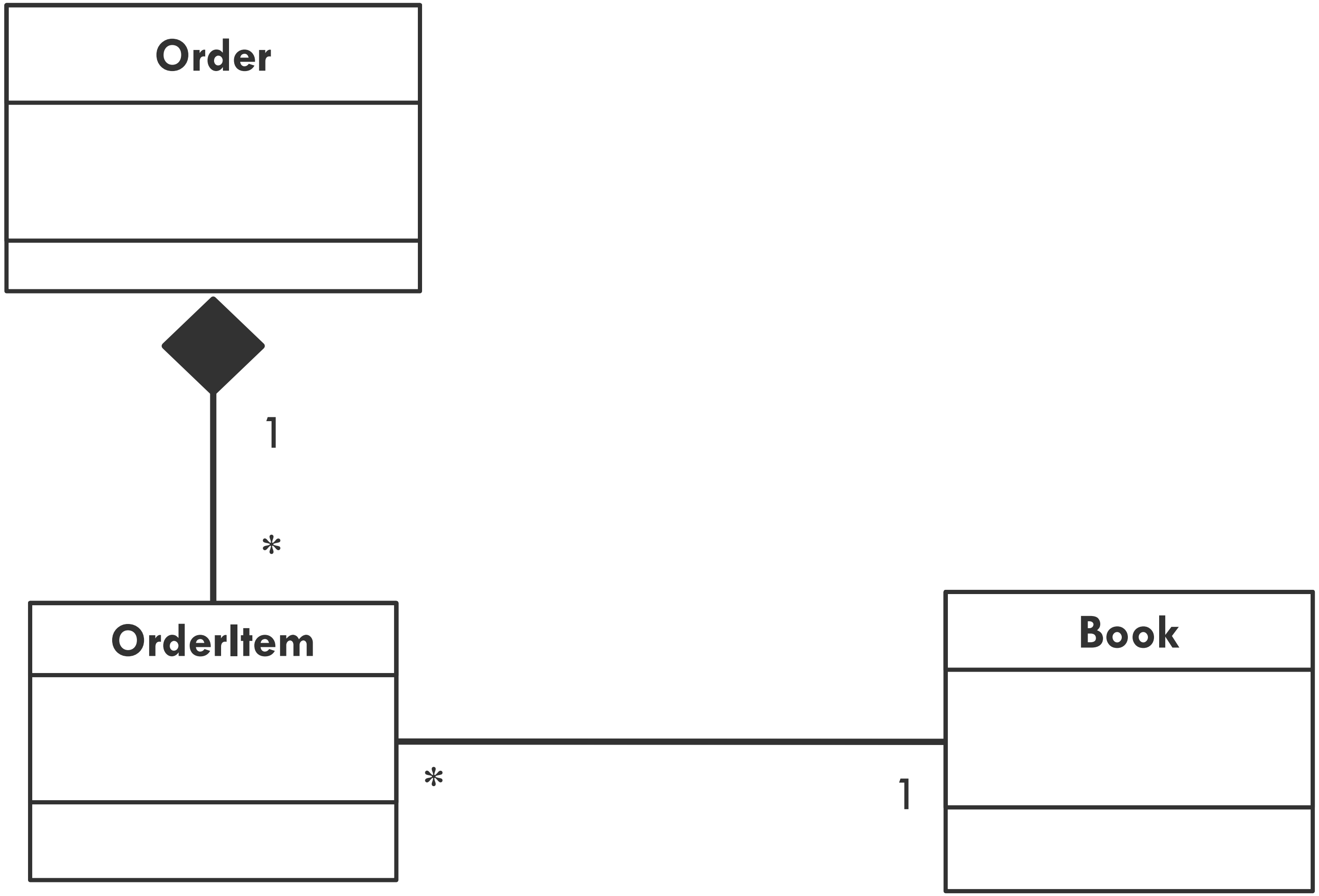
```
public class Group {
    public String name;
    public ArrayList<Contact>contacts;

    // CONSTRUCTOR

    Group(String name){
        this.name=name;
        contacts=new ArrayList<Contact>();
    }

    // ADD EXISTING CONTACT TO A GROUP

    public void addContact(Contact contact){
        contacts.add(contact);
    }
}
```

Composition
Composite Aggregation

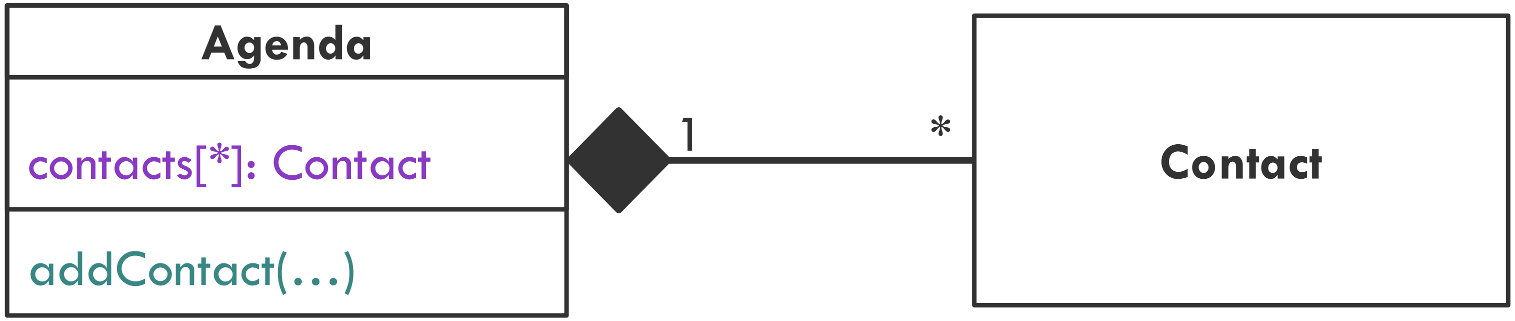


```
public class Patient{
            String name
            MedicalRecord record;
            Patient(){
                        record=new(MedicalRecord);
            }
}
```

Composition
Composite Aggregation

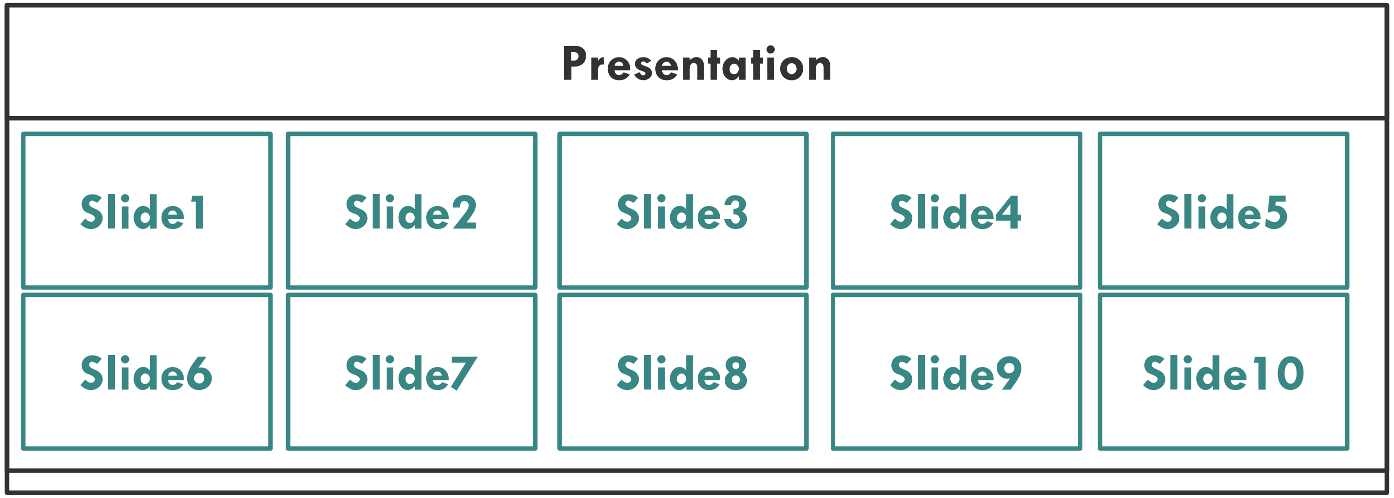


Composition
Composite Aggregation



```
// ADD A NEW CONTACT TO THE AGENDA
 public class Agenda {
   public ArrayList<Contact>contacts;
         public void addContact(String firstName, String lastName, String telephone){
              Contact newContact=new Contact(firstName, lastName, telephone);
              contacts.add(newContact);
         }
}
```

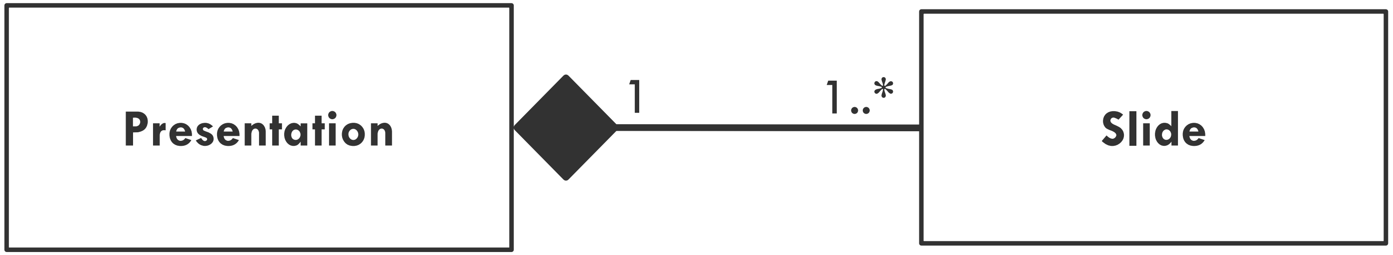
Composition
Composite Aggregation



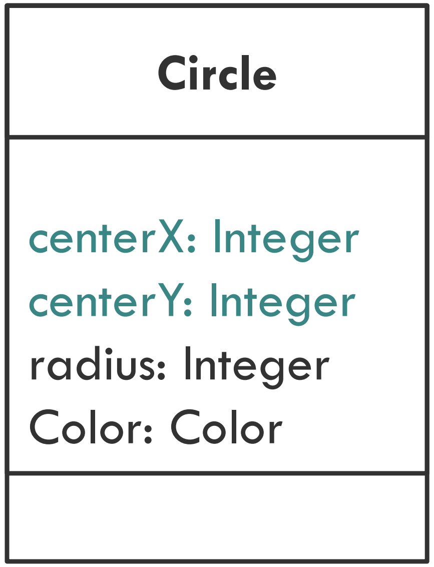
Presentation

Slide1 Slide2 Slide3 Slide4 Slide5

Slide6 Slide7 Slide8 Slide9 Slide10

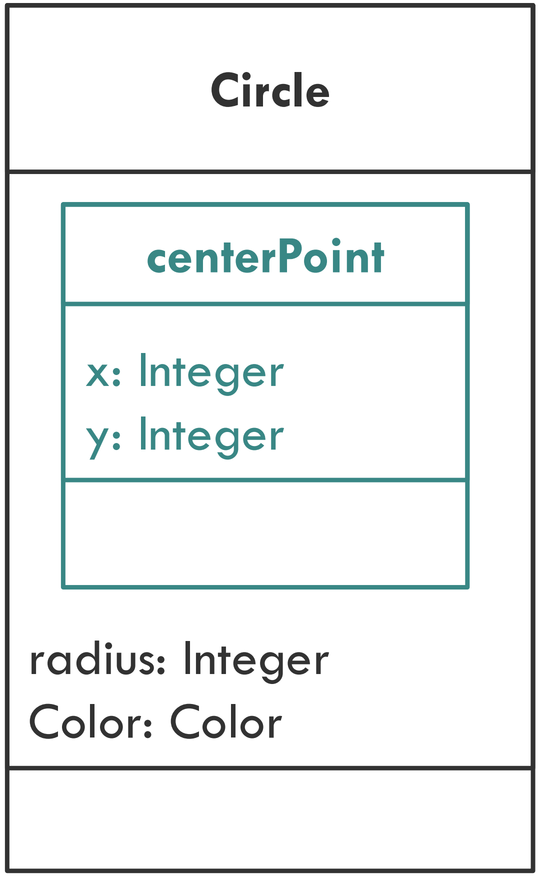


Composition
Composite Aggregation



Circle

centerX: Integer
centerY: Integer
radius: Integer
Color: Color

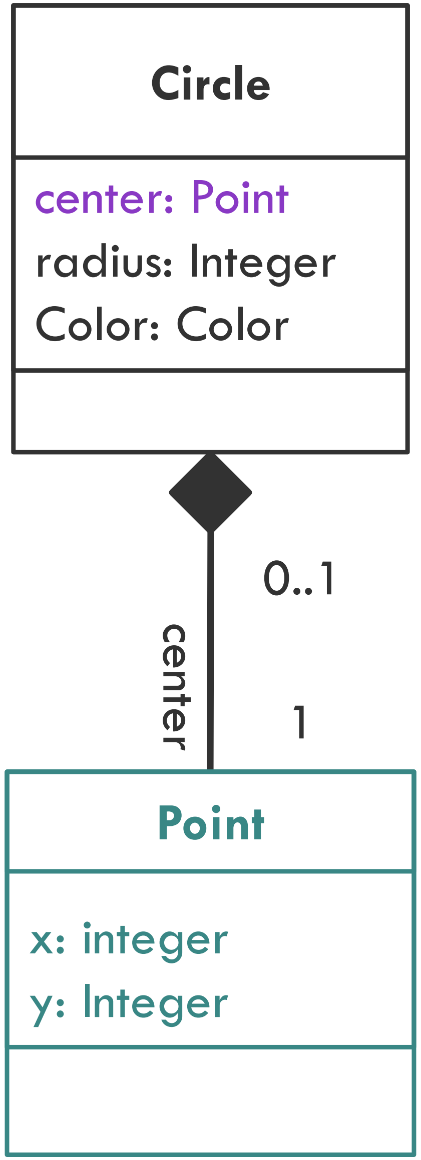


Circle

centerPoint

x: Integer
y: Integer

radius: Integer
Color: Color



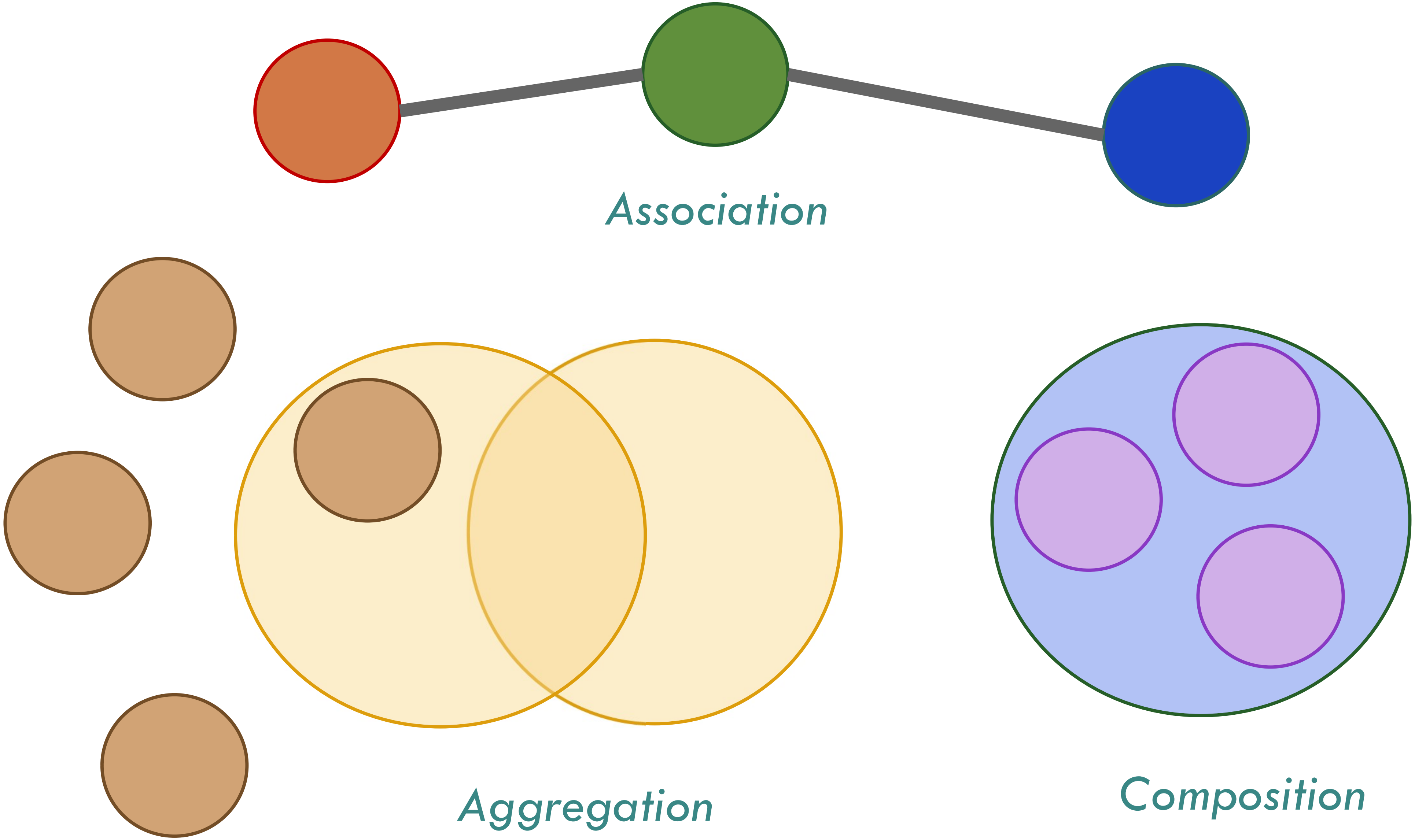
Circle

center: Point
radius: Integer
Color: Color

Point

x: integer
y: Integer

Association, aggregation and composition



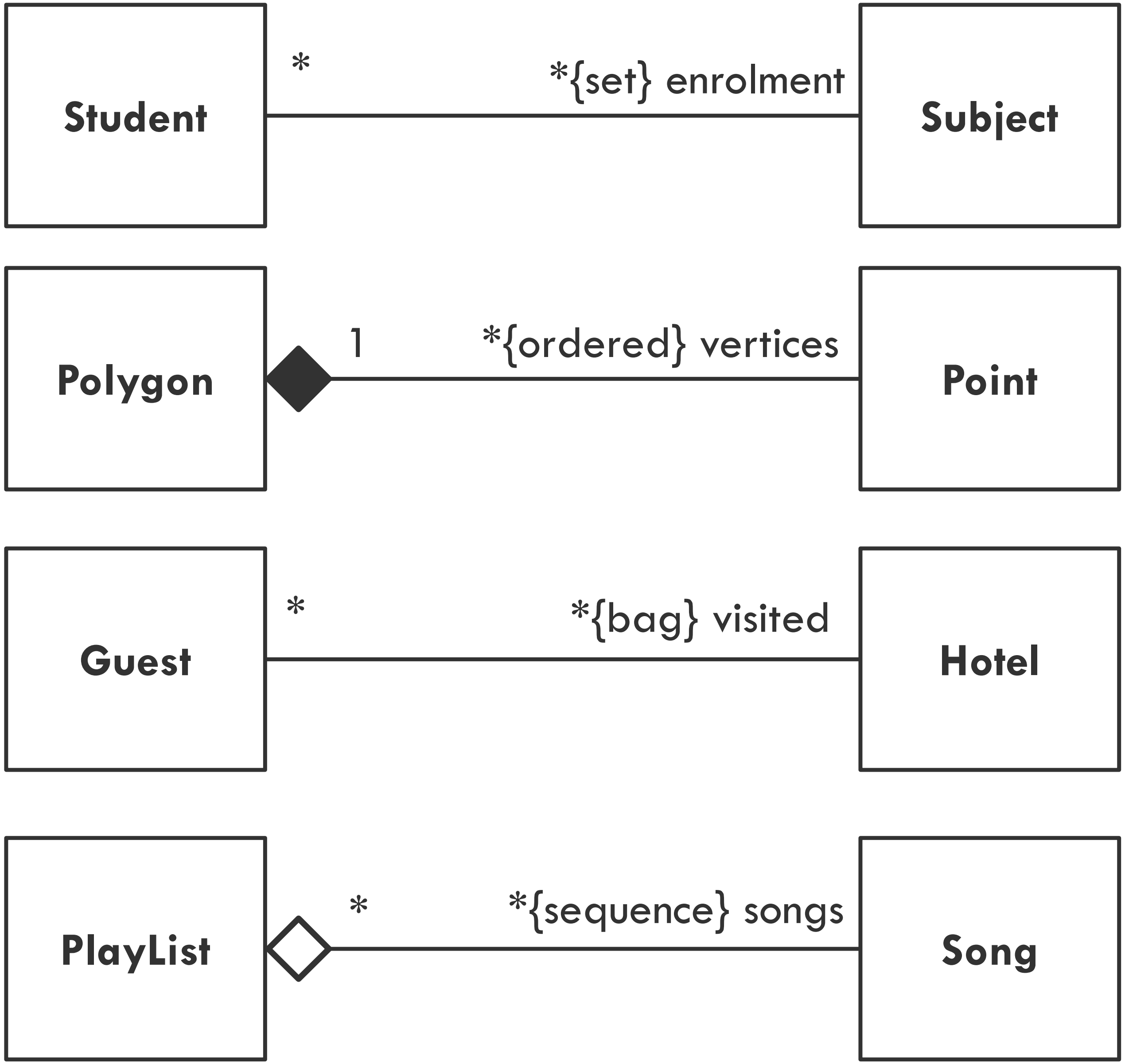
Association

Aggregation

Composition

Collections

Order and repetition in associations



Student * — *{set} enrolment Subject

Polygon 1 — *{ordered} vertices Point

Guest * — *{bag} visited Hotel

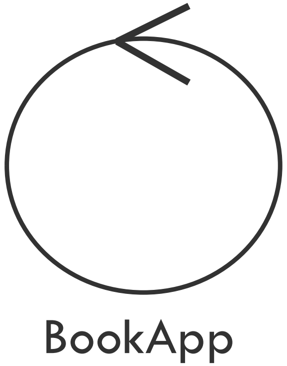
PlayList * — *{sequence} songs Song

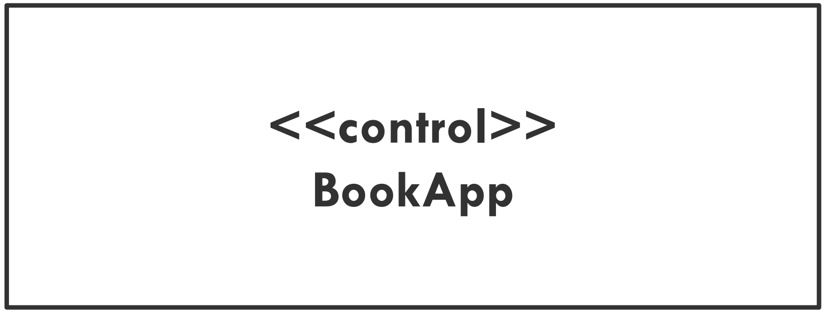
Order	Repetition
✗	✗
✓	✗
✗	✓
✓	✓

(default)

System controller

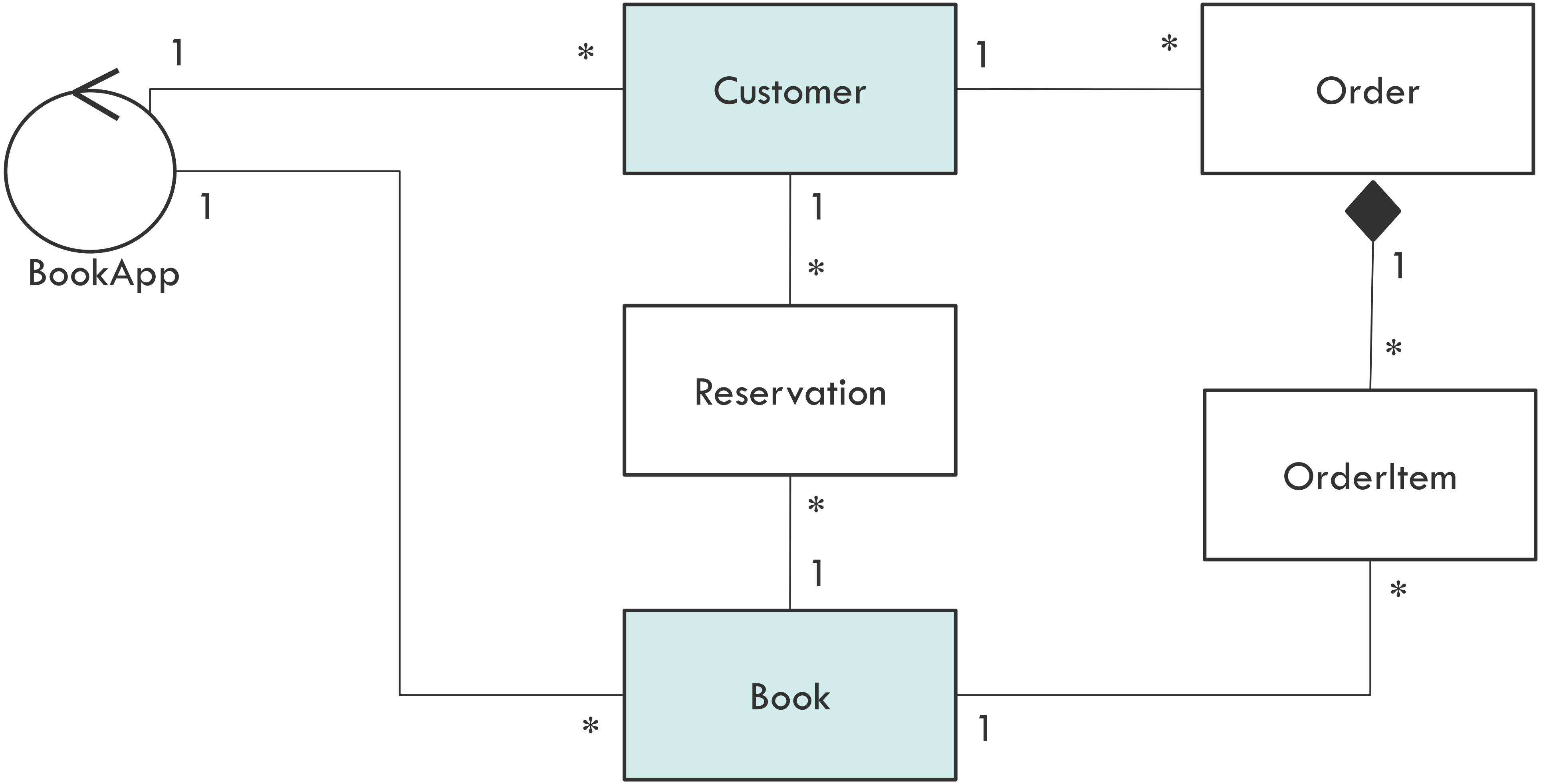
System controller / Façade controller





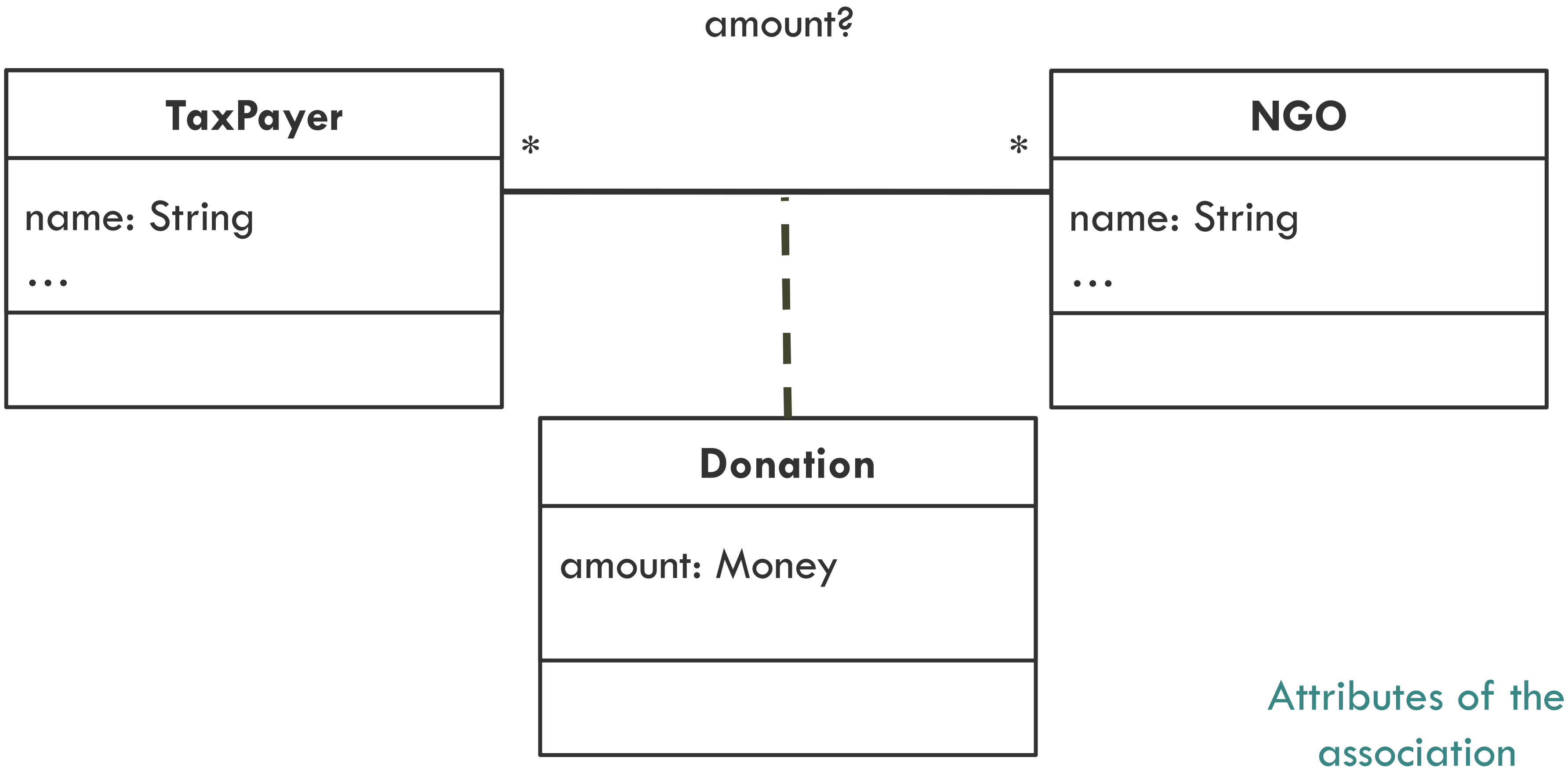
System controller

System controller / Façade controller



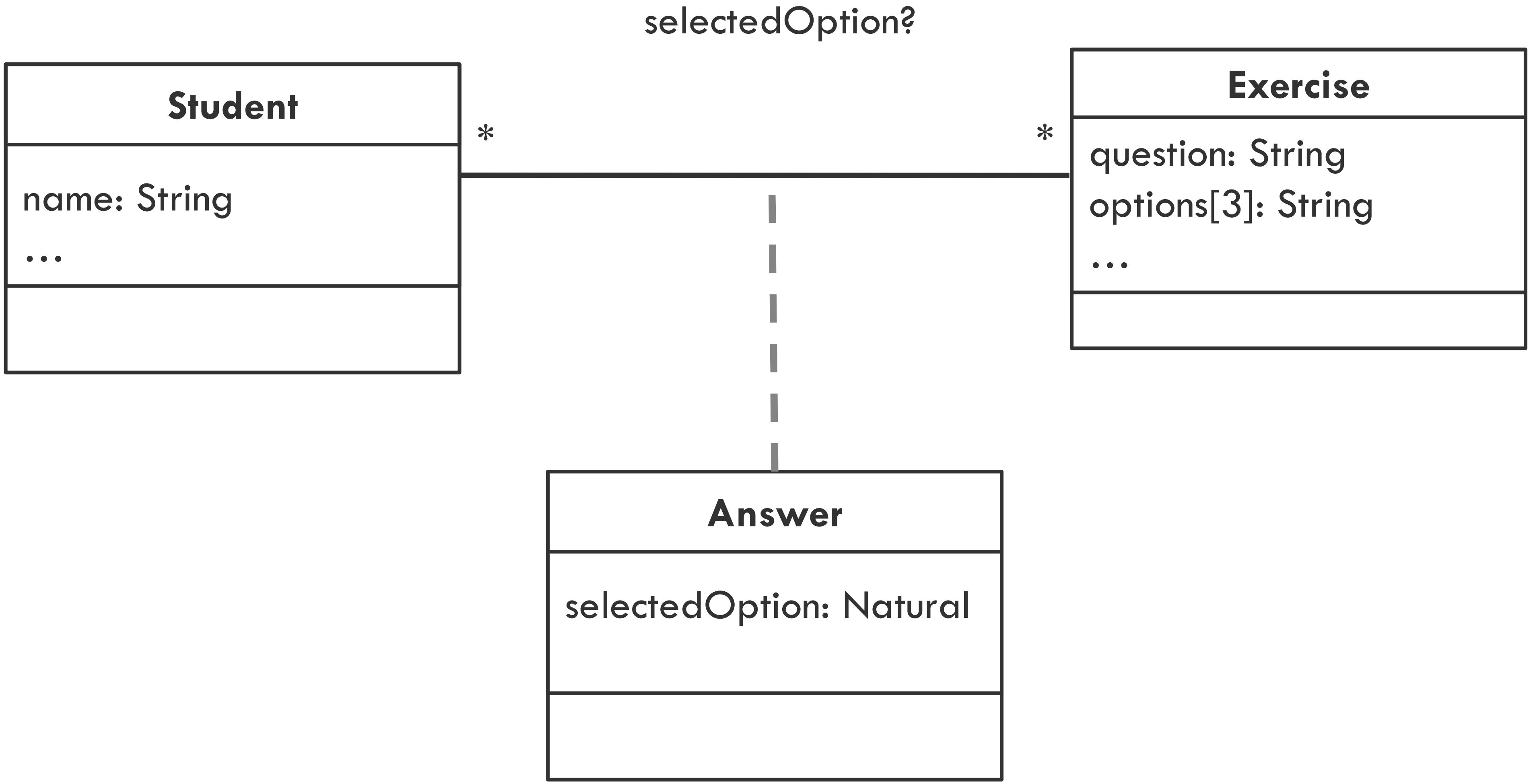
Independent vs dependent concepts

Association classes



Attributes of the association

Association classes



selectedOption?

Student

name: String

…

Exercise

question: String

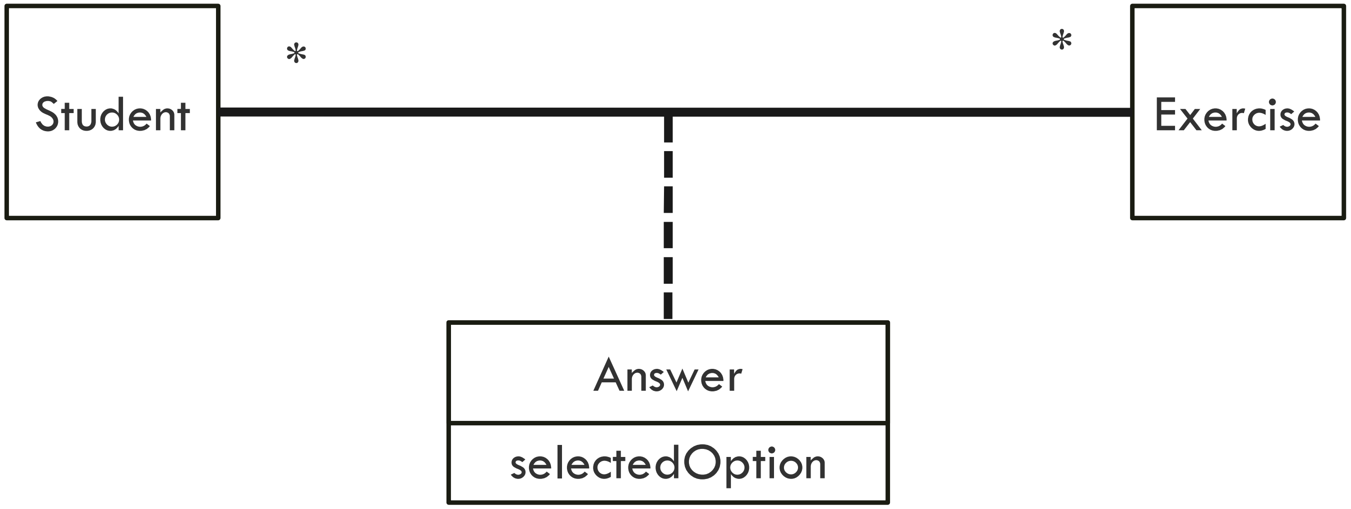
options[3]: String

…

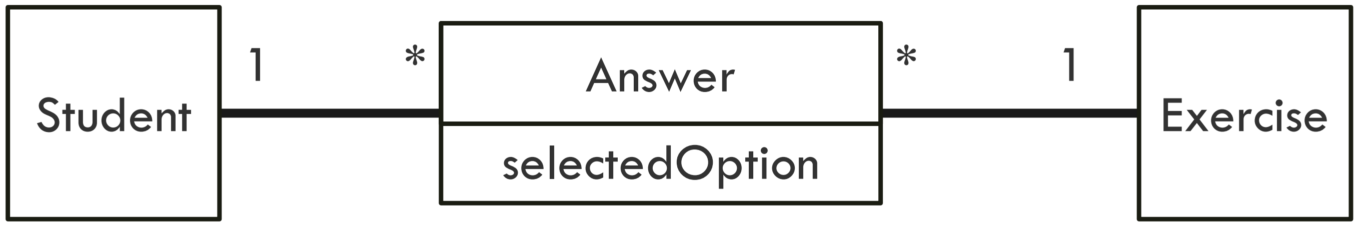
Answer

selectedOption: Natural

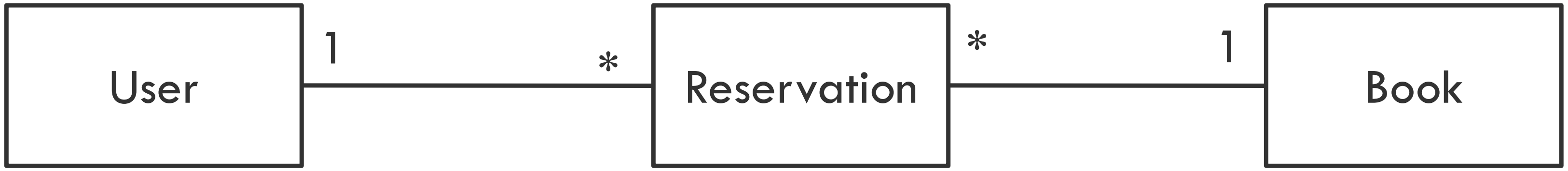
Association classes

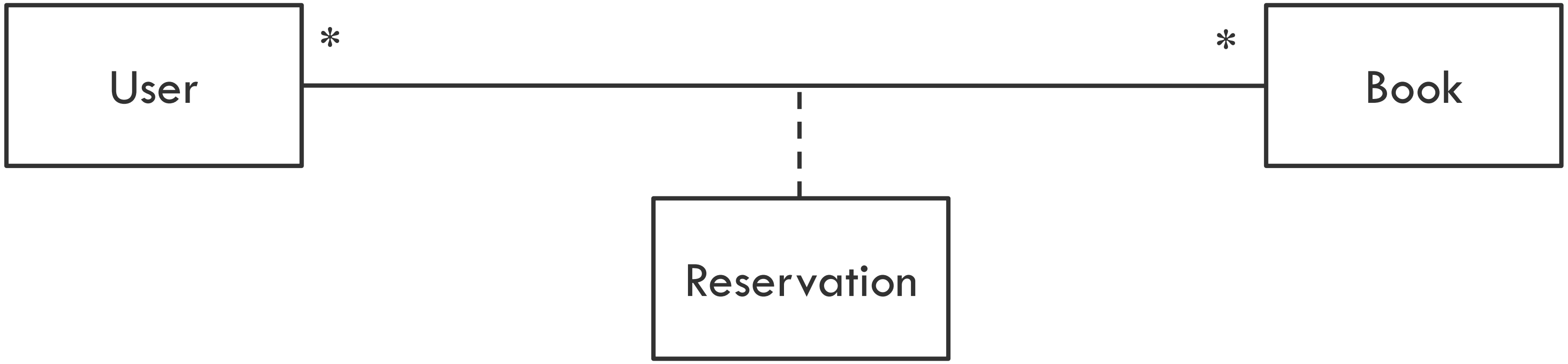


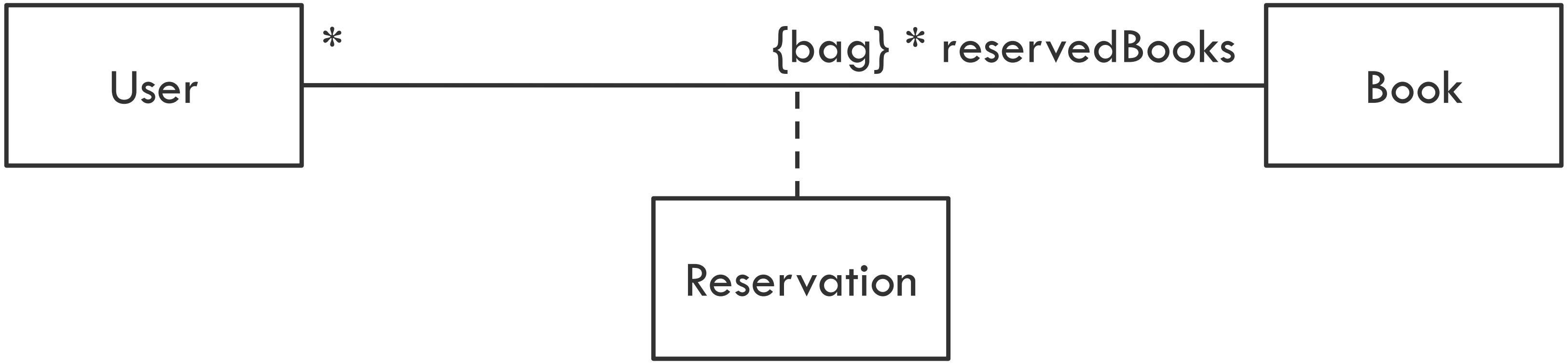
≠



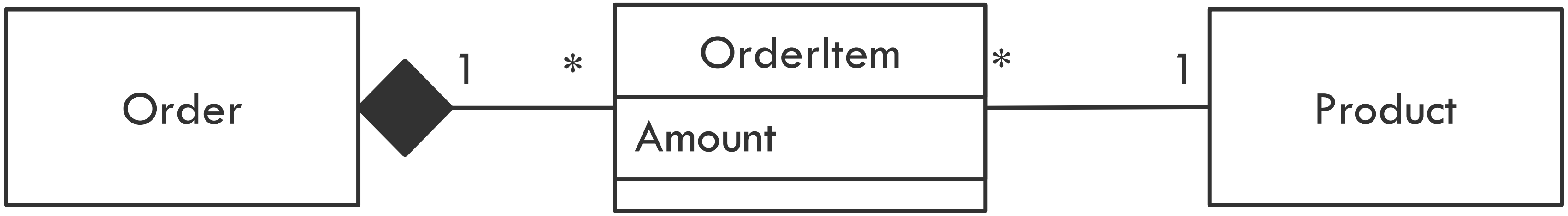
Association classes

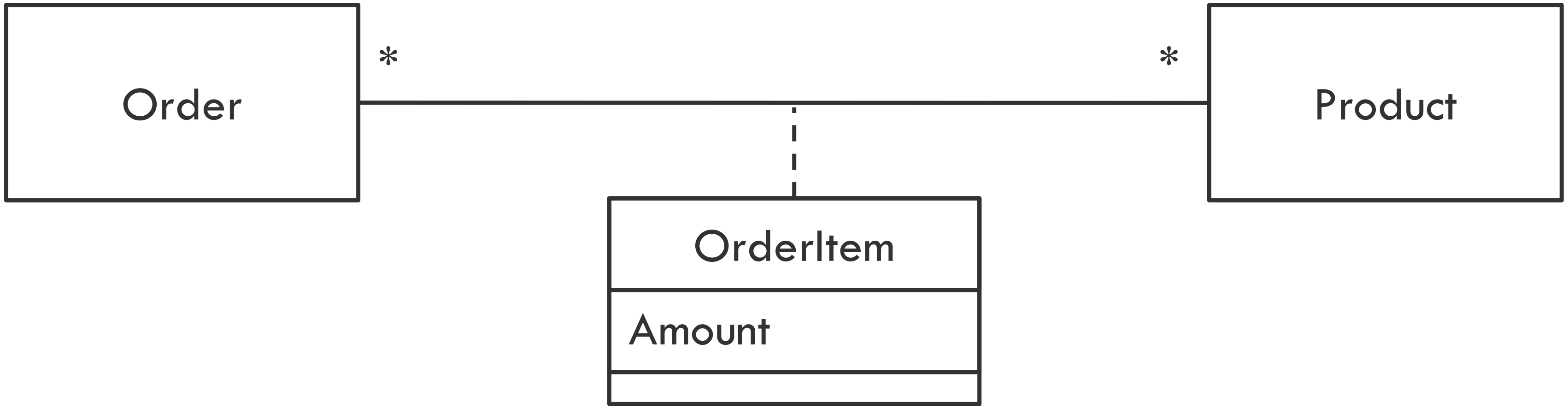


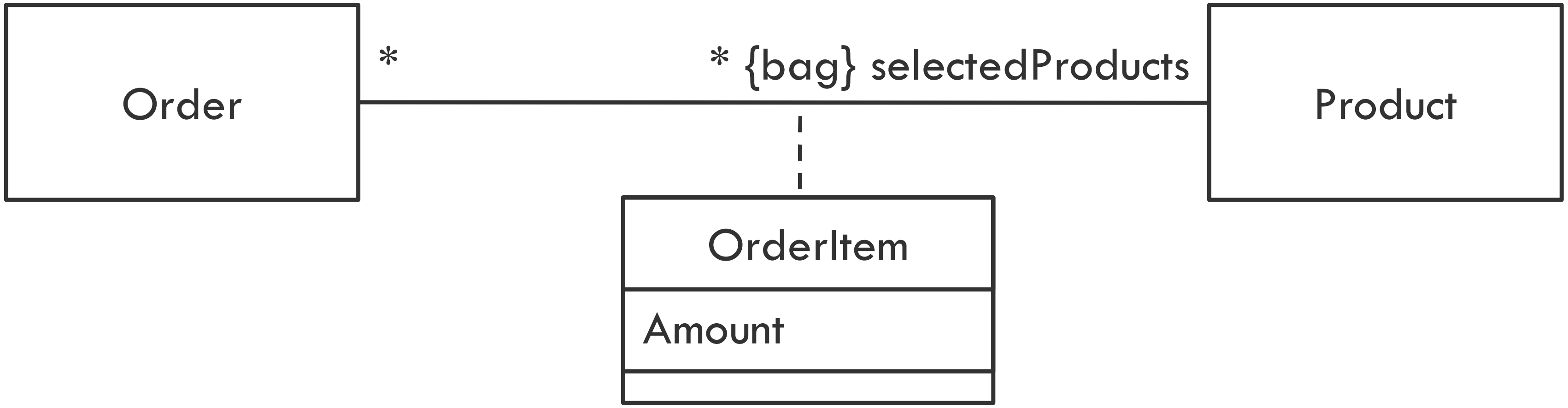




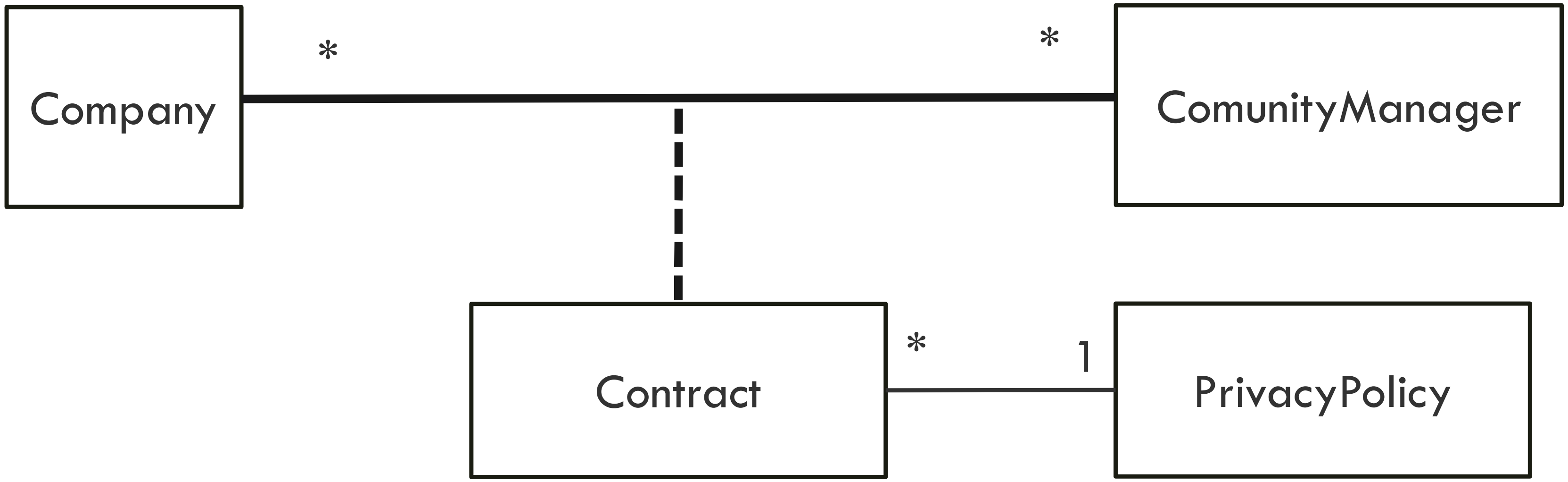
Association classes



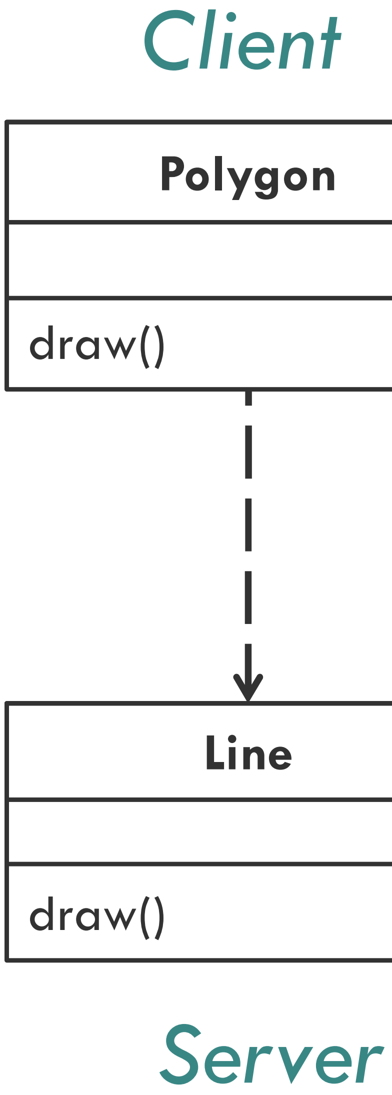




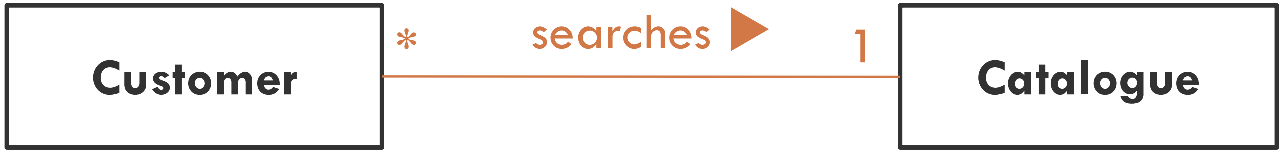
Association classes



Dependency
Client and Server



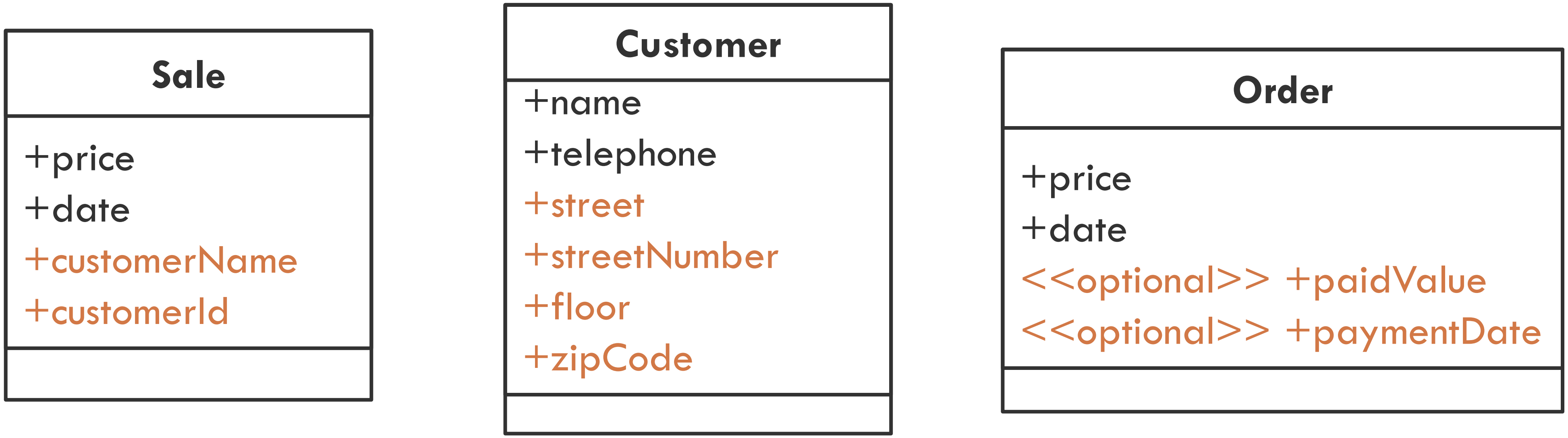
Frequent mistakes



Customer

* searches ▶ 1

Catalogue



Sale

+price
+date
+customerName
+customerId

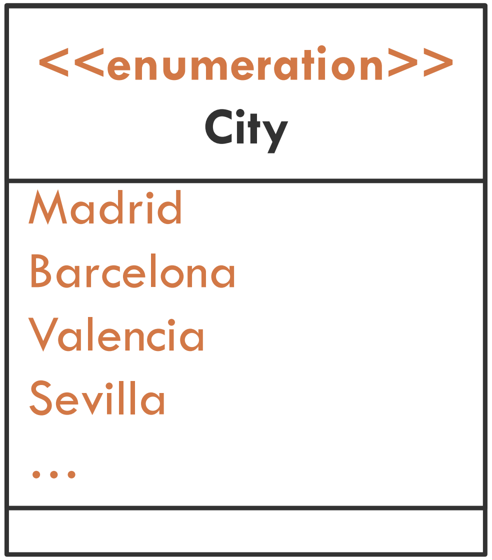
Customer

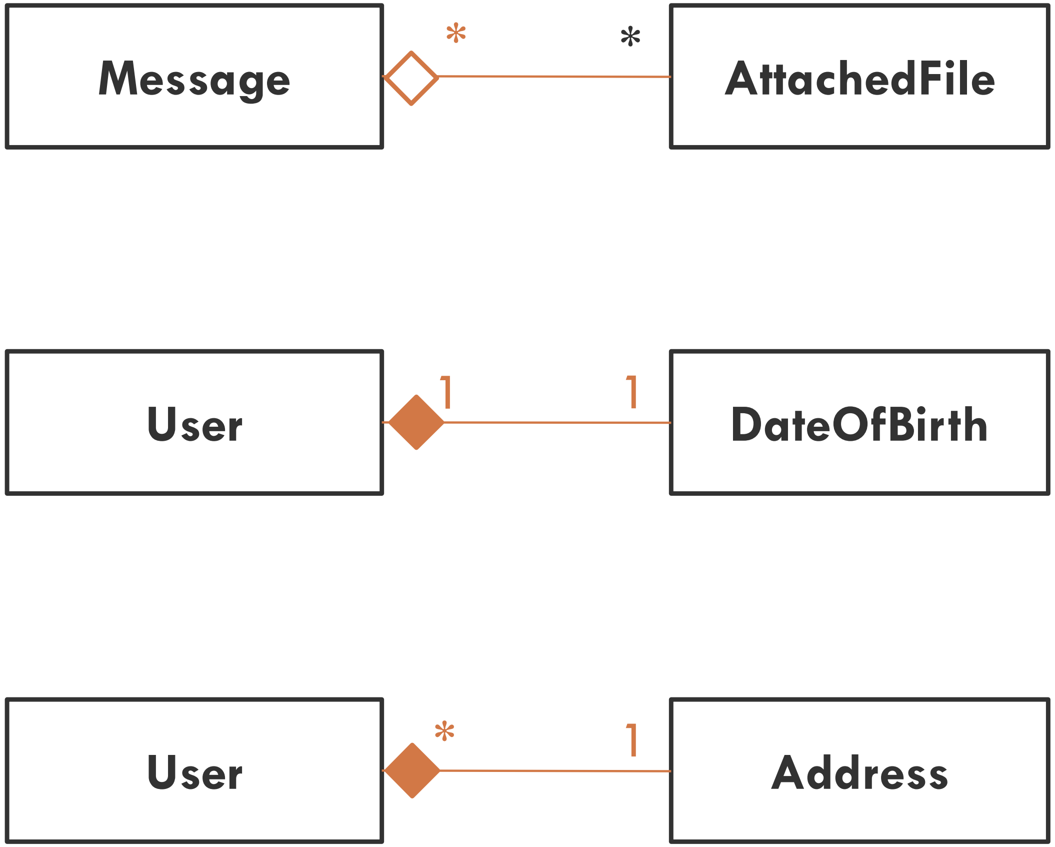
+name
+telephone
+street
+streetNumber
+floor
+zipCode

Order

+price
+date
<<optional>> +paidValue
<<optional>> +paymentDate

Frequent mistakes ❌





Bibliography

- *OMG Unified Modeling Language Specification (Version 2.5.1)* https://www.omg.org/spec/UML/2.5.1/ . *OMG, December 2017*

- *El Lenguaje Unificado de Modelado. Manual de Referencia / Unified Modeling Languge Reference Manual, 2nd Edition*

  James Rumbaugh, Ivar Jacobson y Grady Booch. Addison Wesley, 2004

- *Object-Oriented Analysis and Design for Information Systems. Modeling with UML, OCL and FML*

  Raul Sidnei Wazlawick. Morgan Kaufmann (ed.), 2014

- *UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition*

  Martin Fowler. Addison Wesley Professional, 2003

- *UML™ Bible*

  Tom Pender. John Wiley & Sons 2003