DATA MODELLING IN ENGINEERING

MODELING IN UML – PART I -

Ana Inês Oliveira aio@fct.unl.pt

2024 - 2025

Contents

- **Introduction to UML**

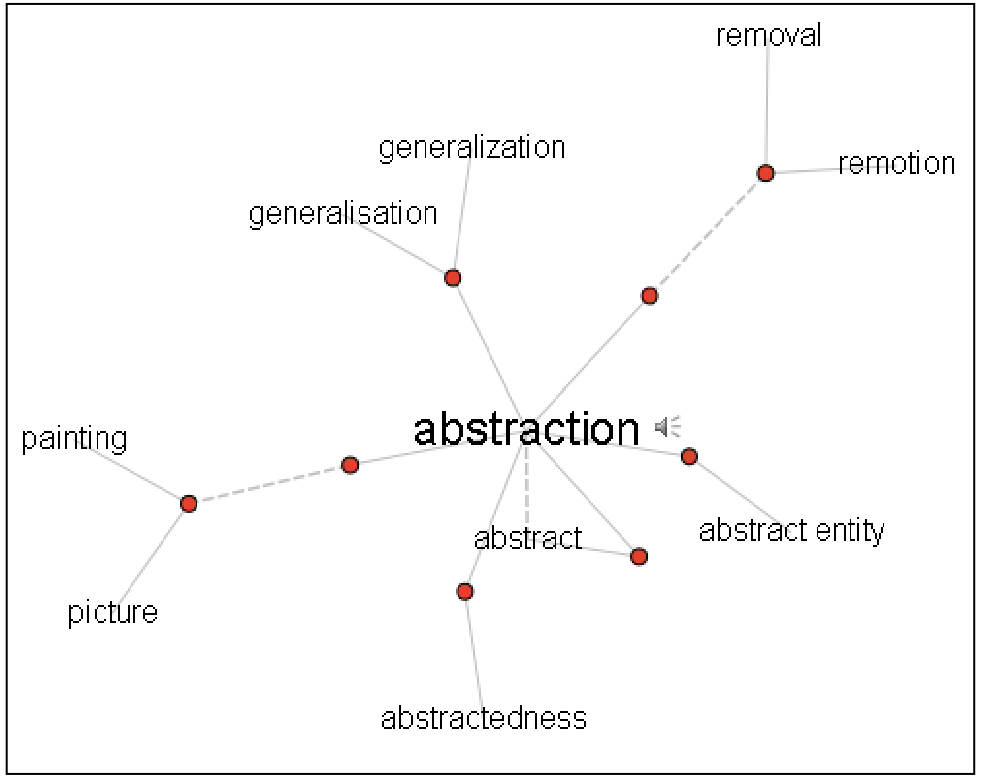
    - Abstraction / Models
    - UML Basic Elements
        - UML Things
            - Structural Things
            - Behaviour Things
            - Grouping Things
            - Annotational Things
        - UML Relationships
        - UML Diagrams
            - UML Structure Diagrams
            - UML Behaviour Diagrams
    - Class Diagram

https://www.uml.org/

Introduction to UML

Abstraction

- What is abstraction?
- Abstraction is a conceptual process where general rules and concepts are derived from the usage and classification of specific examples
- Abstraction is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.

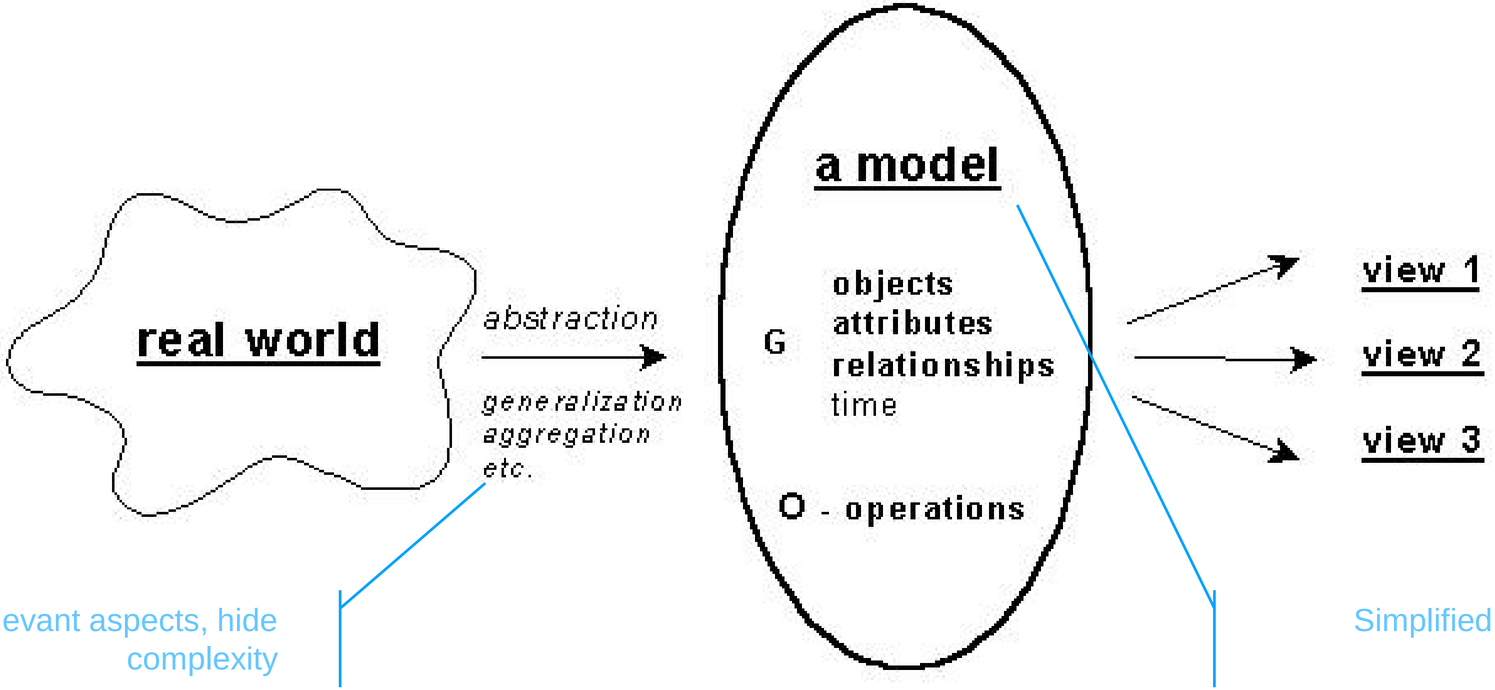


http://www.visualthesaurus.com/app/view

Introduction to UML

Role of models in understanding, designing, and communicating

Abstraction



Focus on relevant aspects, hide complexity

Simplified representations of reality

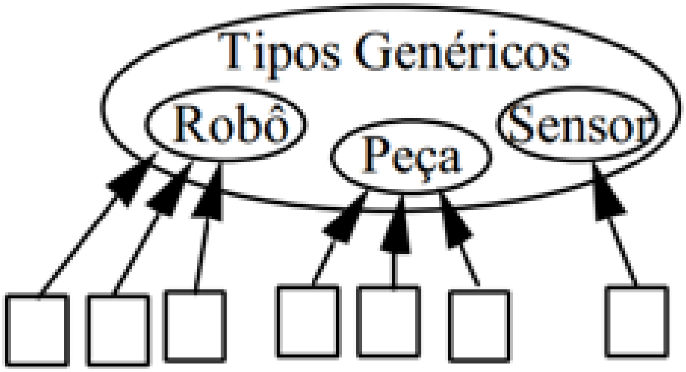
… but, how we abstractly represent things depends on our

INTENTIONS AND GOALS

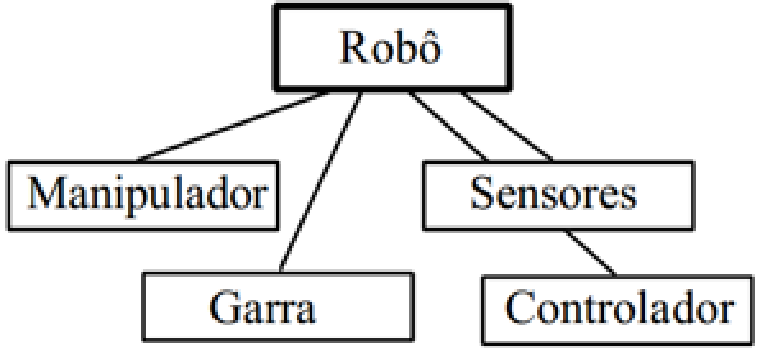
Introduction to UML

Abstraction

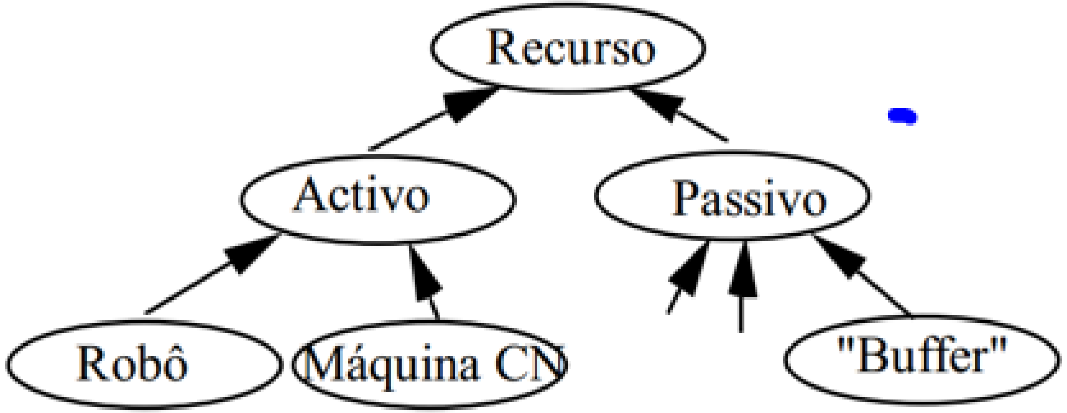
- Abstraction mechanisms



Classification



Aggregation / Composition



Generalization

Introduction to UML

Models are used for:

- Communication between humans
- Communication between humans and computers
- Communication between computers for information transmission, processing, interpretation, feature extraction…

- To achieve an abstract and simplified representation of a problem to reduce its complexity and allow adequate studying / resolution / …

- It is also used to perform calculations, simulate and for behavior prediction.

- …/…

Introduction to UML

What.... Why UML?

- Modeling Language

- The OMG's Unified Modeling Language™ (UML®) helps you specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements. (You can use UML for business modeling and modeling of other non-software systems too.)

- Modeling is the designing of software applications before coding. Modeling is an Essential Part of large software projects, and helpful to medium and even small projects as well.

[http://www.uml.org/what-is-uml.htm]

Introduction to UML

What…. Why UML?

- A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other - a conceptual model can be defined as a model which is made of concepts and their relationships.

- UML plays an important role in defining different perspectives of a system. These perspectives are:
  - Design
  - Implementation
  - Process
  - Deployment

The center is the Use Case view which connects all these four.

A Use Case represents the functionality of the system

Introduction to UML

What…. Why UML?

- **Design** of a system consists of classes, interfaces, and collaboration.

  UML provides class diagram, object diagram to support this.

- **Implementation** defines the components assembled together to make a complete physical system.

  UML component diagram is used to support the implementation perspective.

- **Process** defines the flow of the system.

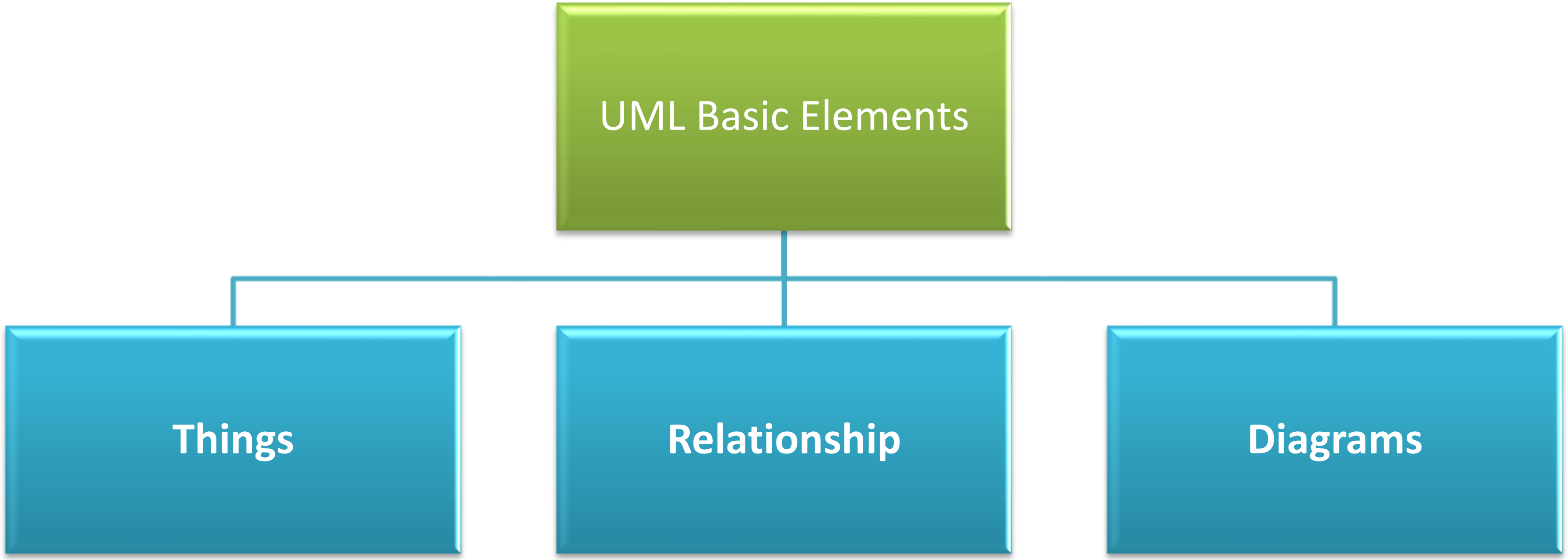
  The same elements as used in Design are also used to support this perspective.

- **Deployment** represents the physical nodes of the system that forms the hardware.

  UML deployment diagram is used to support this perspective.

Introduction to UML

UML Basic Elements



UML Basic Elements

Things

Relationship

Diagrams

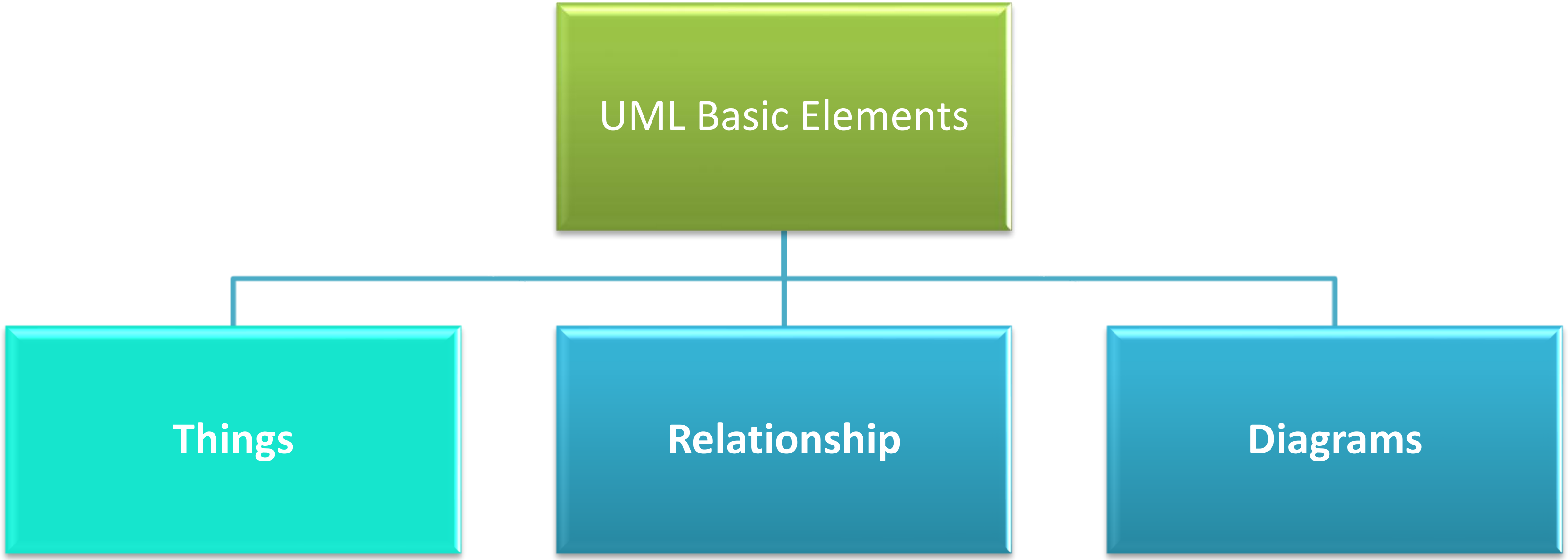
Structural things
Behavioural things
Grouping things
Annotational things

Dependency
Association
Generalization
Realization

Class diagram
Object diagram
Use Case diagram
Sequence diagram
(…)

Introduction to UML

UML Basic Elements



UML Basic Elements

Things

Relationship

Diagrams

Structural things
Behavioural things
Grouping things
Annotational things

Dependency
Association
Generalization
Realization

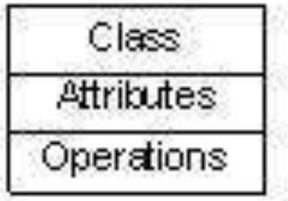
Class diagram
Object diagram
Use Case diagram
Sequence diagram
(…)

Introduction to UML

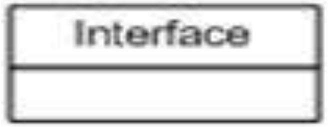
UML Things: Structural things

Structural things define the static part of the model.

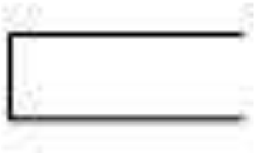
They represent the physical and conceptual elements.



- **Class** – represents a set of objects having similar responsibilities.



- **Interface** – defines a set of operations, which specify the responsibility of a class.



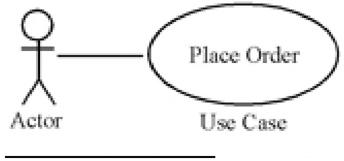
- **Collaboration** – defines an interaction between elements.

Introduction to UML

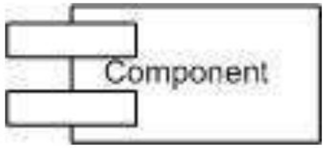
UML Things: Structural things

Structural things define the static part of the model.

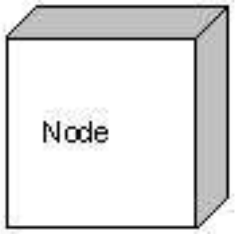
They represent the physical and conceptual elements.



- **Use case** – represents a set of actions performed by a system for a specific goal.



- **Component** – describes the physical part of a system.

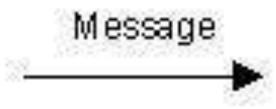


- **Node** – can be defined as a physical element that exists at run time.

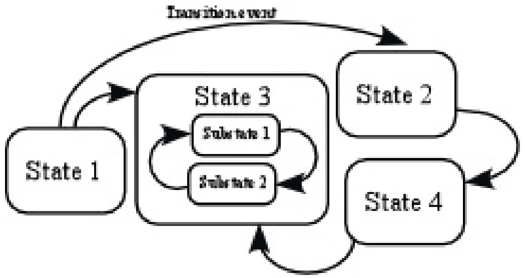
Introduction to UML

UML Things: Behavioural things

A behavioral thing consists of the dynamic parts of UML models.



- **Interaction** – defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.

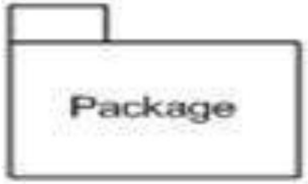


- **State machine** – useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change

Introduction to UML

UML Things: Grouping things

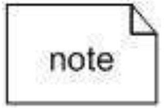
Grouping things can be defined as a mechanism to group elements of a UML model together.



- **Package** − is the only one grouping thing available for gathering structural and behavioral things.

UML Things: Annotational things

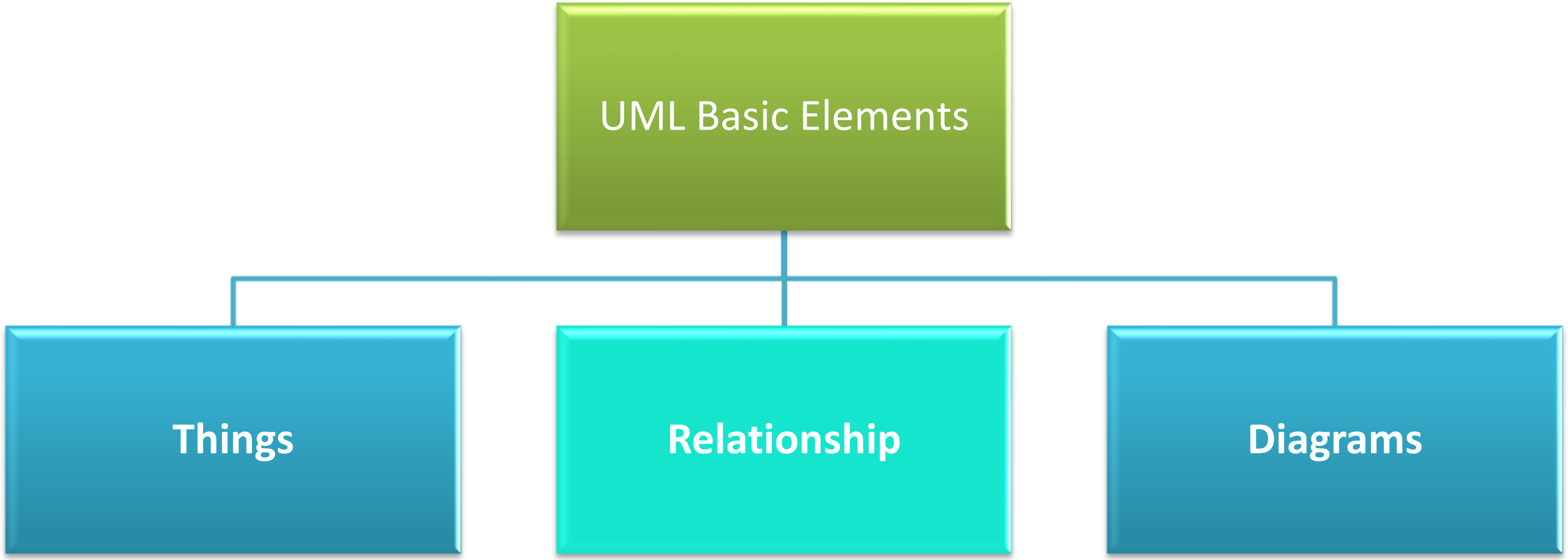
Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements.



- **Note** - It is the only one Annotational thing available. A note is used to render comments, constraints, etc. of an UML element.

Introduction to UML

UML Basic Elements



UML Basic Elements

Things

Relationship

Diagrams

Structural things
Behavioural things
Grouping things
Annotational things

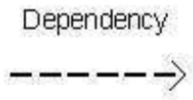
Dependency
Association
Generalization
Realization

Class diagram
Object diagram
Use Case diagram
Sequence diagram
(…)

Introduction to UML

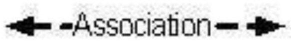
UML Relationships

Relationship is another important building block of UML. It shows how the elements are associated with each other, and this association describes the functionality of an application.



A uses B

- **Dependency** − relationship between two things in which change in one element also affects the other.



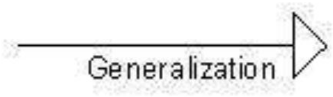
A is connected to B

- **Association** – a set of links that connects the elements of a UML model. It also describes how many objects are taking part in that relationship.

Introduction to UML

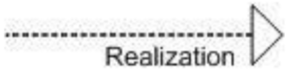
UML – relationship

Relationship is another important building block of UML. It shows how the elements are associated with each other, and this association describes the functionality of an application.



A is a type of B

- **Generalization** – relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the world of objects.

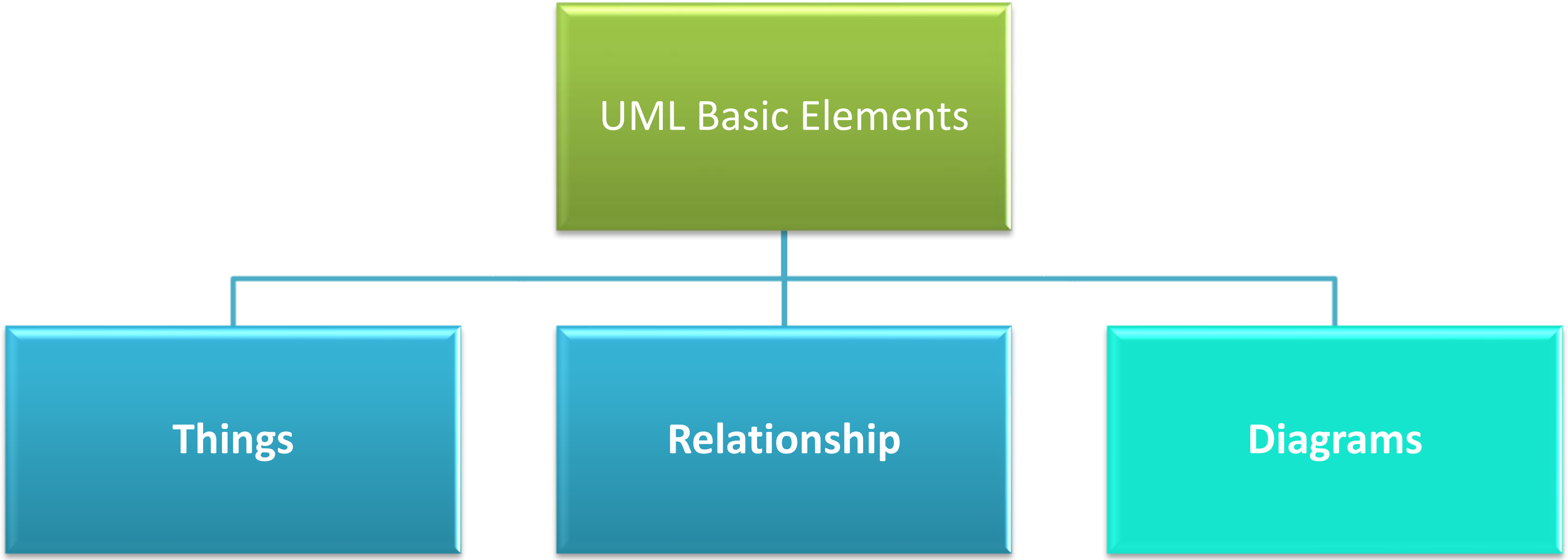


A implements B

- **Realization** – relationship in which two elements are connected. One element describes some responsibility, which is not implemented, and the other one implements them.

Introduction to UML

UML Basic Elements



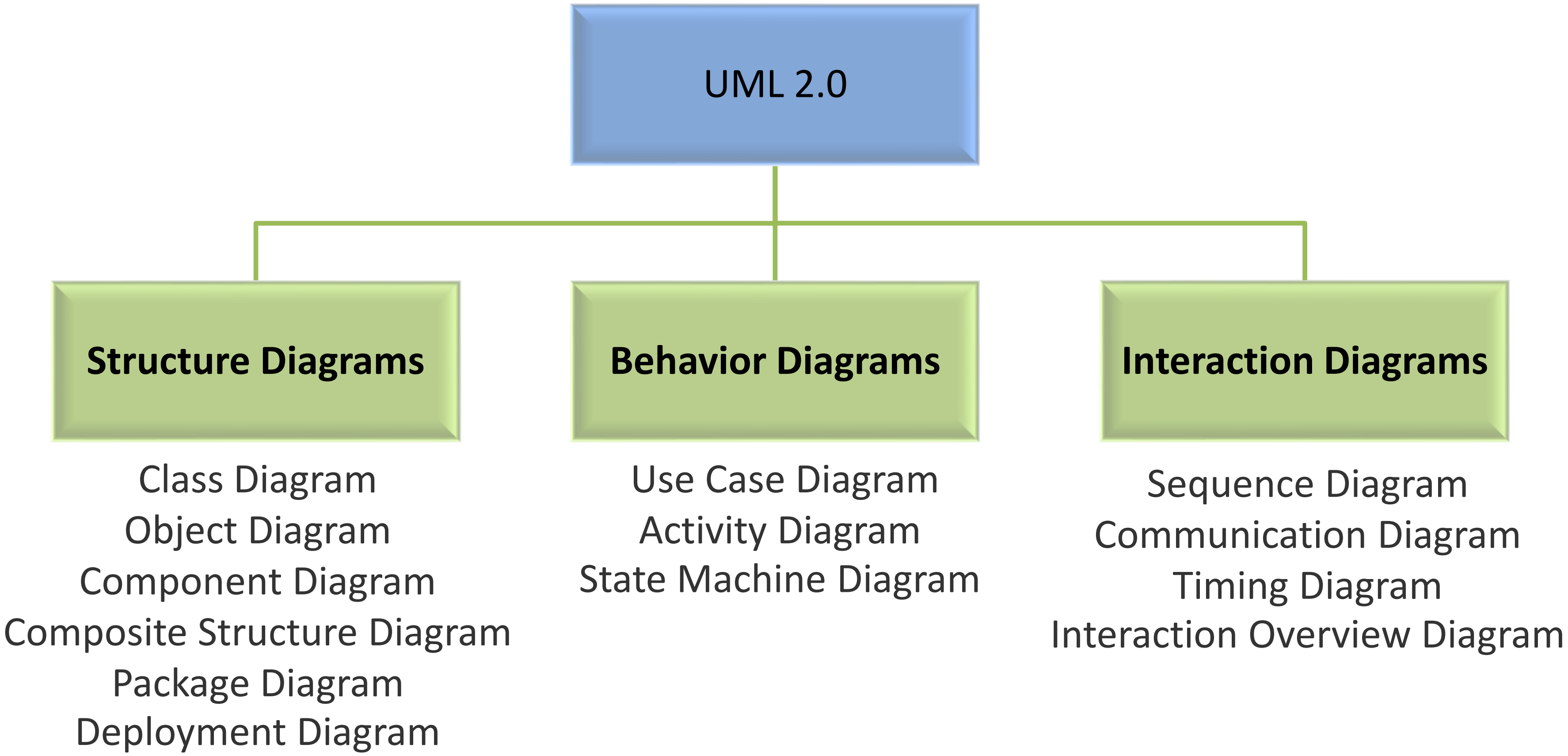
Structural things
Behavioural things
Grouping things
Annotational things

Dependency
Association
Generalization
Realization

Class diagram
Object diagram
Use Case diagram
Sequence diagram
(…)

Introduction to UML

UML Diagrams:



UML 2.0

Structure Diagrams

Class Diagram
Object Diagram
Component Diagram
Composite Structure Diagram
Package Diagram
Deployment Diagram

Behavior Diagrams

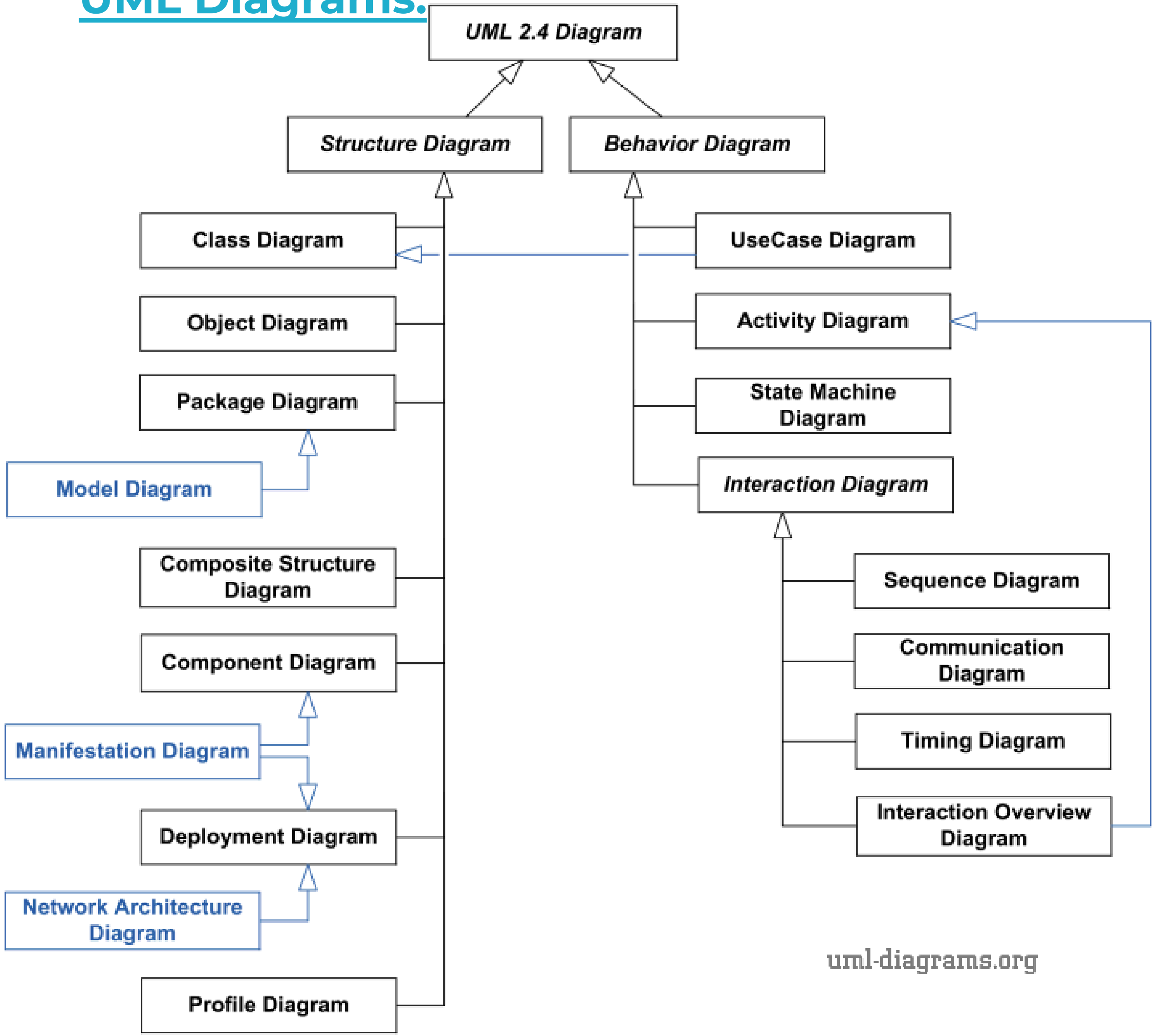
Use Case Diagram
Activity Diagram
State Machine Diagram

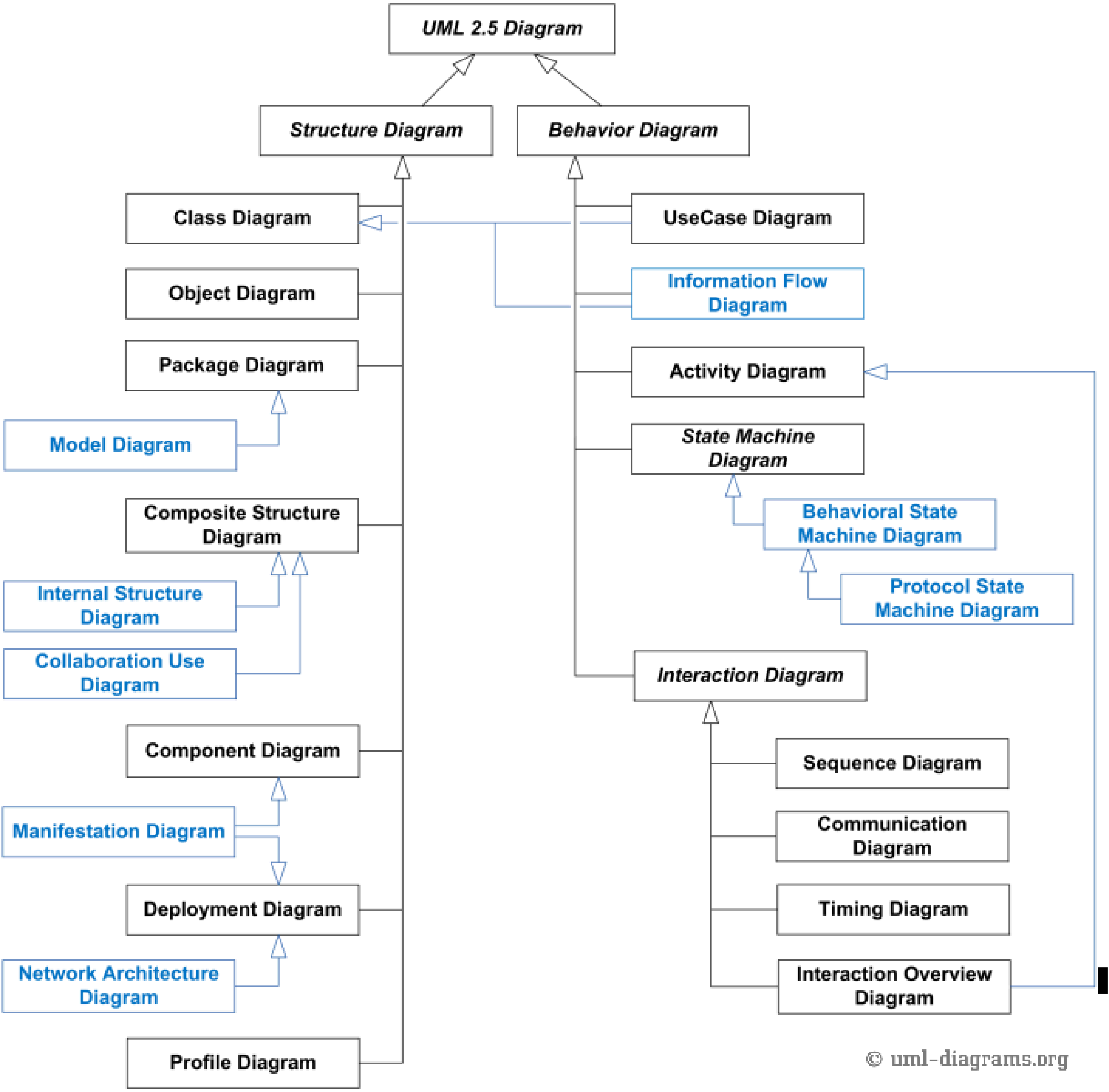
Interaction Diagrams

Sequence Diagram
Communication Diagram
Timing Diagram
Interaction Overview Diagram

Introduction to UML

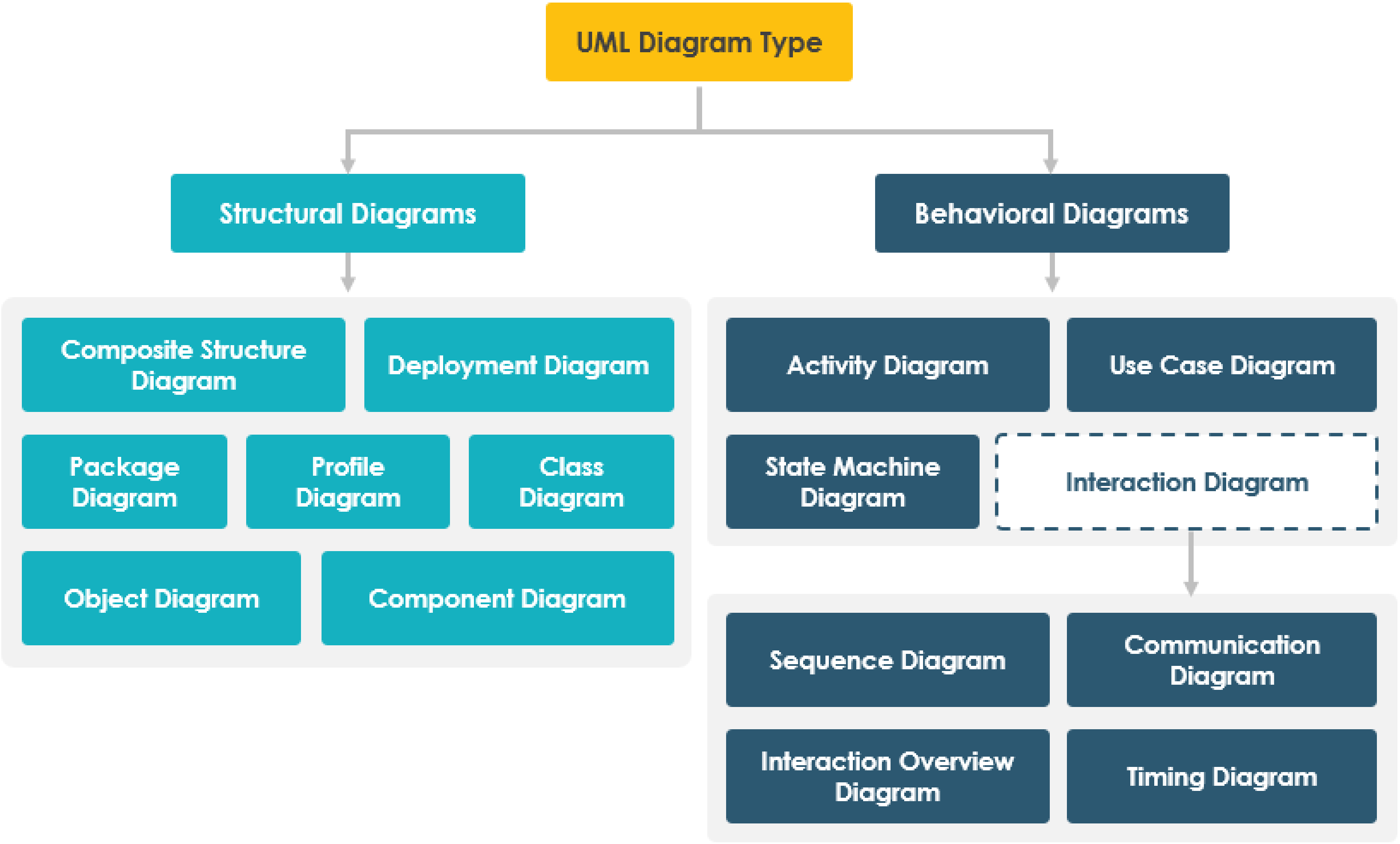
UML Diagrams:





Introduction to UML

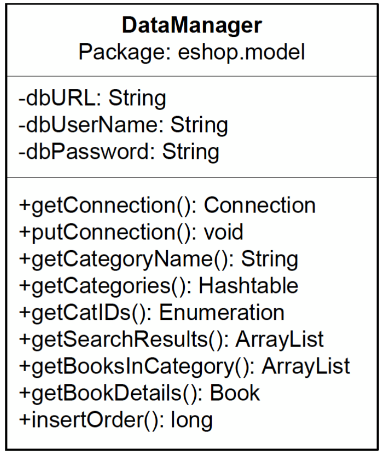
UML Diagrams:



https://www.visual-paradigm.com/guide/uml-unified-modeling-language/overview-of-the-14-uml-diagram-types/

Introduction to UML

UML Structure Diagrams: captures the static features of a system


DataManager
Package: eshop.model
-dbURL: String
-dbUserName: String
-dbPassword: String
+getConnection(): Connection
+putConnection(): void
+getCategoryName(): String
+getCategories(): Hashtable
+getCatIDs(): Enumeration
+getSearchResults(): ArrayList
+getBooksInCategory(): ArrayList
+getBookDetails(): Book
+insertOrder(): long

Class Diagram

Object Diagram

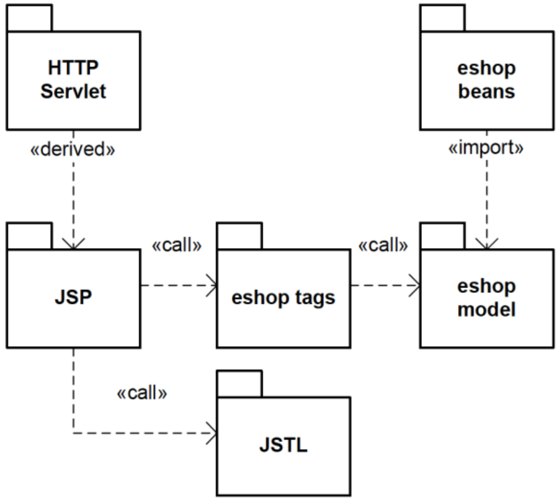
Component Diagram

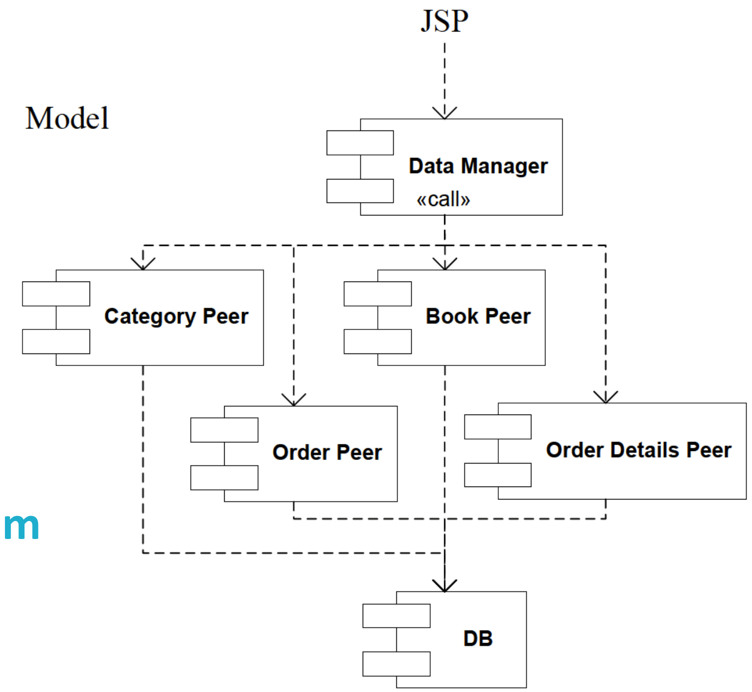
Composite Structure Diagram

Package Diagram

Deployment Diagram

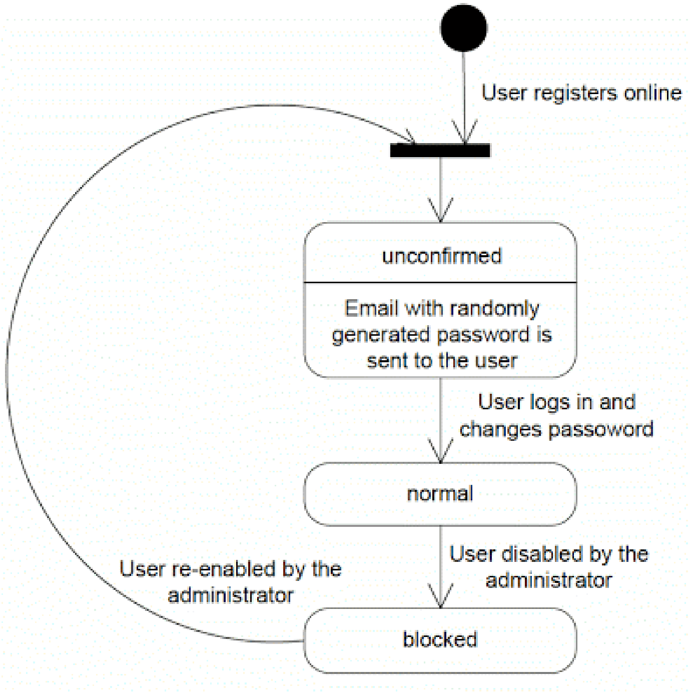
etc.

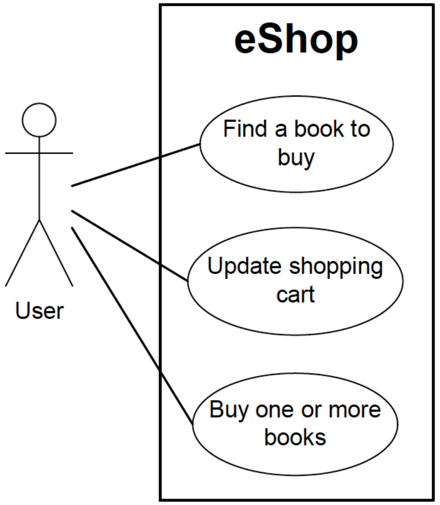




Introduction to UML

UML Behaviour Diagrams: describes the interaction in the system


User registers online
unconfirmed
Email with randomly generated password is sent to the user
User logs in and changes passoword
normal
User disabled by the administrator
User re-enabled by the administrator
blocked

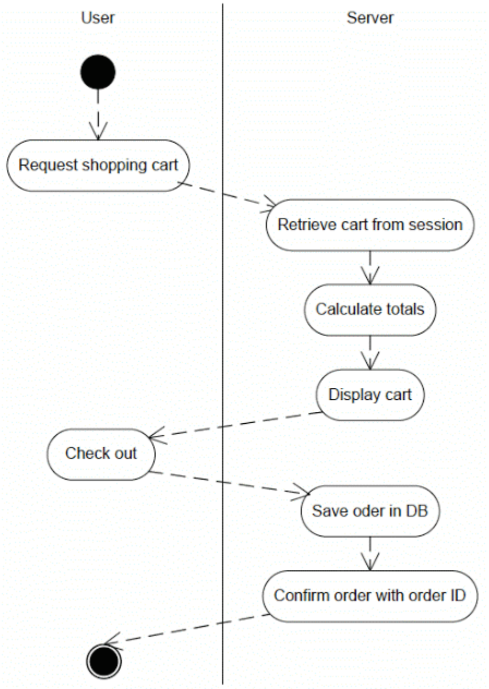


Use Case Diagram

Activity Diagram

State Machine Diagram

etc.



Interaction Diagrams

Sequence Diagram
Communication Diagram
Timing Diagram
Interaction Overview Diagram

Introduction to UML

Start with UML:

- **Select a methodology**:

  A methodology formally defines the process that you use to gather requirements, analyze them, and design an application that meets them in every way. There are many methodologies, each differing in some way or ways from the others.

- **Select a UML Development Tool**:

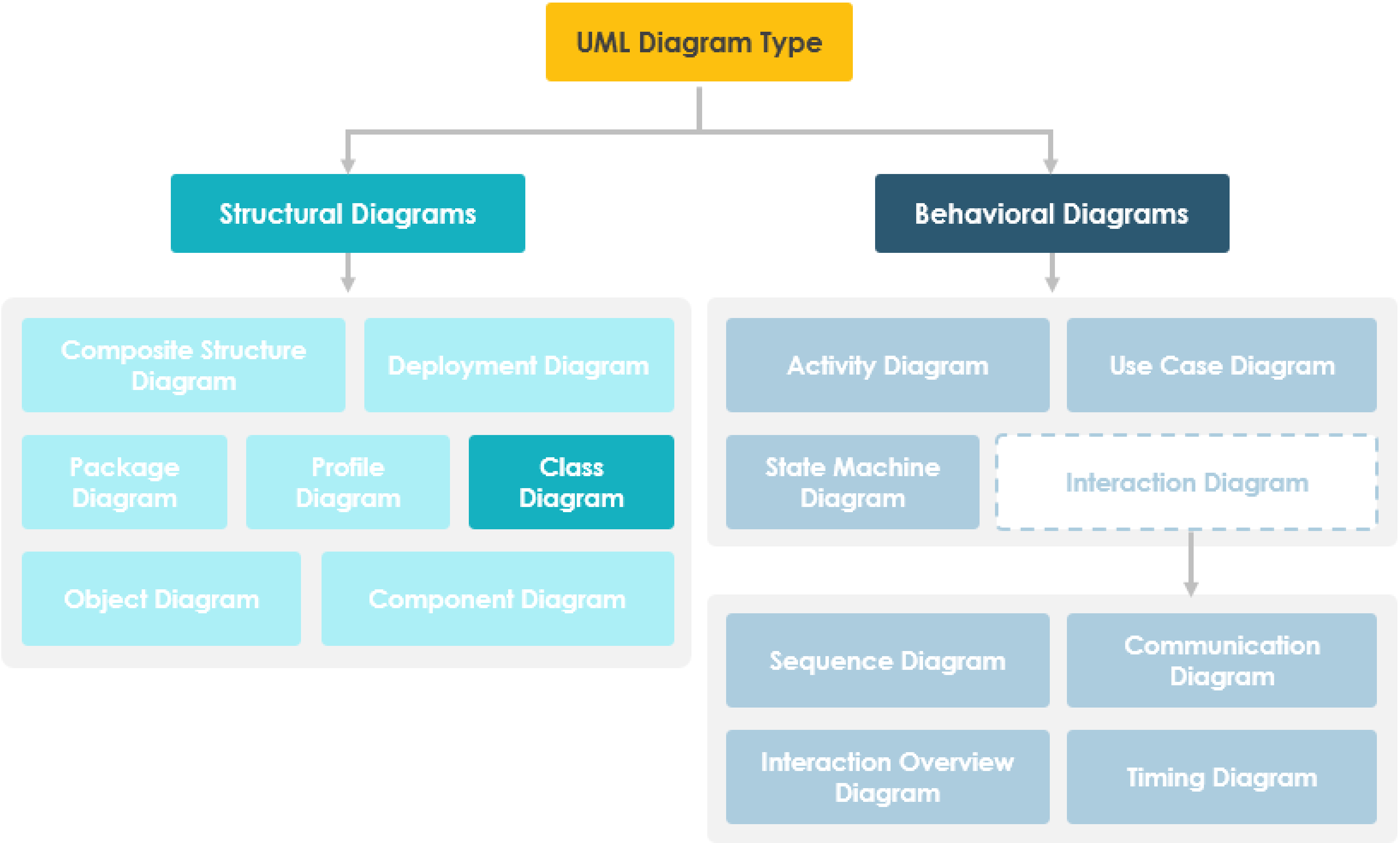
  Because most (although not all) UML-based tools implement a particular methodology, in some cases it might not be practical to pick a tool and then try to use it with a methodology that it wasn't built for.

- **Get Training**

[http://www.uml.org/what-is-uml.htm]

Introduction to UML

Class Diagrams:

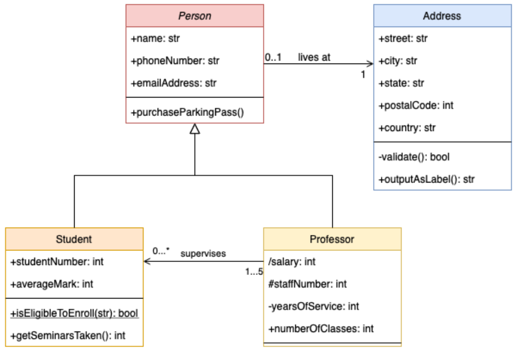


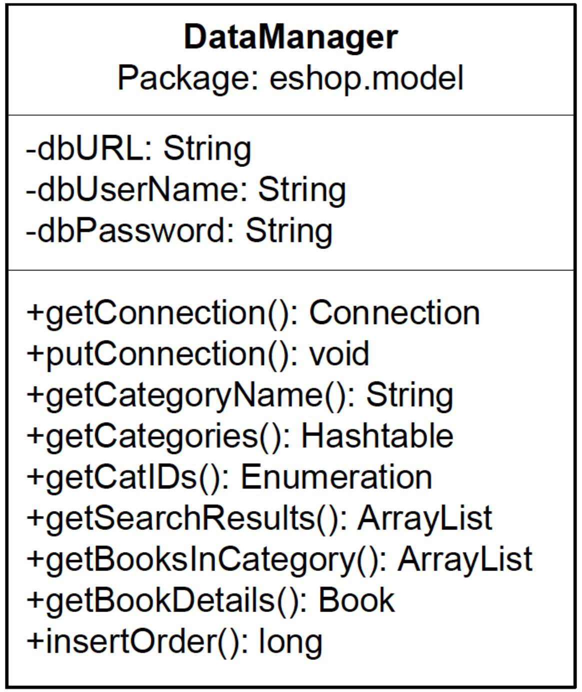
https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

Introduction to UML

Class Diagrams:

type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects





DataManager
Package: eshop.model

-dbURL: String
-dbUserName: String
-dbPassword: String

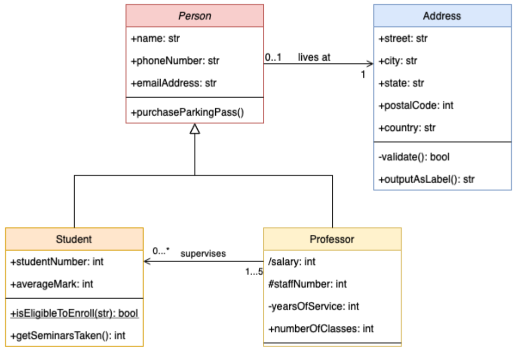
+getConnection(): Connection
+putConnection(): void
+getCategoryName(): String
+getCategories(): Hashtable
+getCatIDs(): Enumeration
+getSearchResults(): ArrayList
+getBooksInCategory(): ArrayList
+getBookDetails(): Book
+insertOrder(): long

Class name

Attributes

Methods / Operations

Introduction to UML

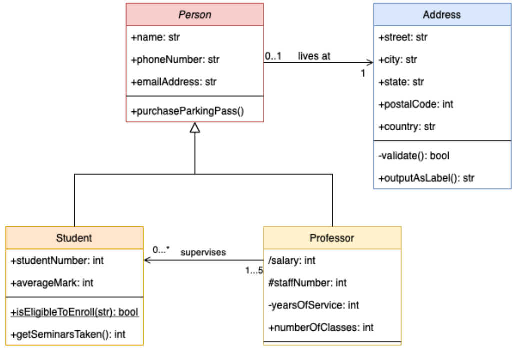


Class Diagrams:

- **Objects**
  - Objects represent an entity and the basic building block.
  - Object is something that takes up space in the real or conceptual world with which somebody may do things (Booch et al . 1999)
  - The objects have :
    - ***Name*** (or ID): The entire object must have a name that will differentiate from other objects in a context (eg. calculator)
    - ***State***: An object has state, which involves the object's properties together with the values of these properties (e.g. connected calculator)
    - ***Operations*** (or behavior):can do something with the object or the object can do something with another object (e.g. calculator does sums)

Introduction to UML

Class Diagrams:



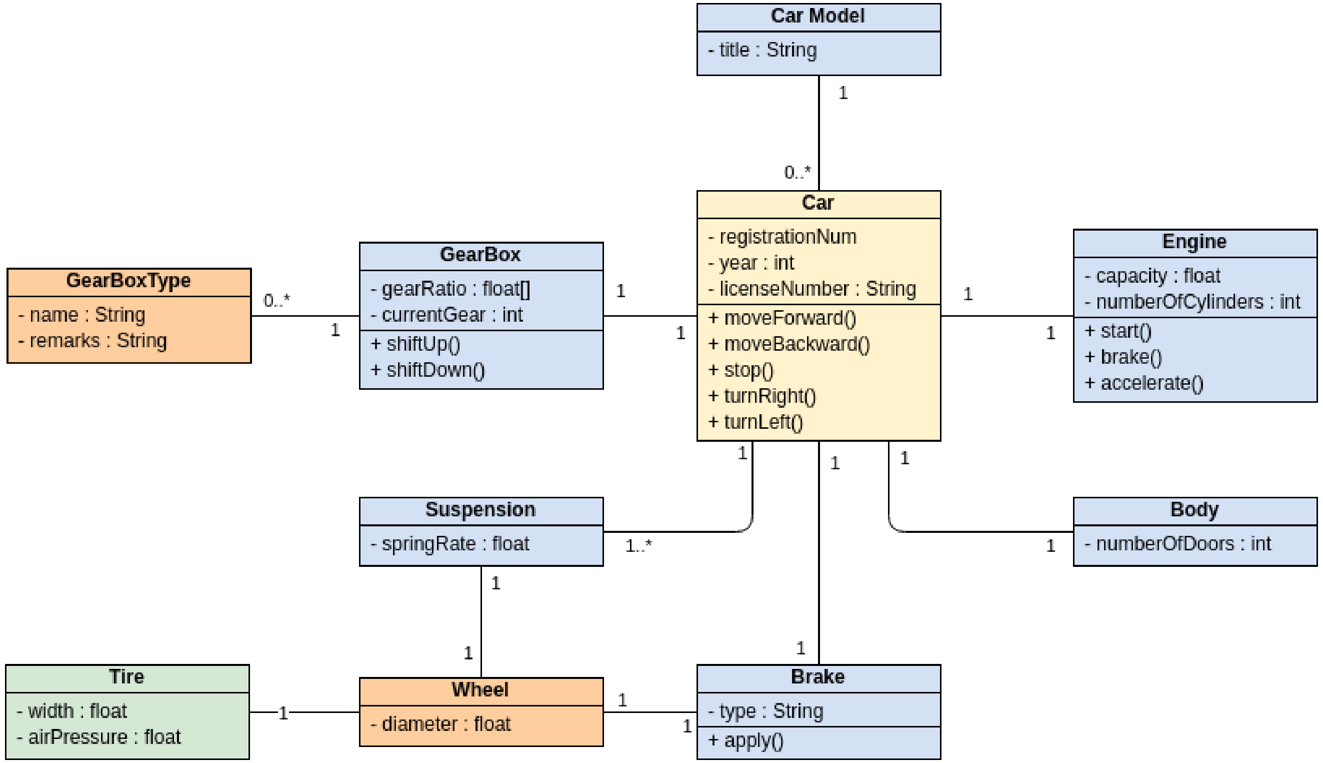
- **Classes**
  - A class is the description of a set of objects that share the same attributes, operations, relationships and semantics. (Eg calculators).
  - Class is the abstraction of an object.
    - Object is an instantiation of a class…
  - Illustrate data models for information systems

Introduction to UML

Class Diagrams:

- Main modeling elements:
  - Classes
  - Relationships (association and generalization)
  - Multiplicity

- Main purposes:
  - Model main objects
  - Model simple collaboration
  - Logical Model of DB



Car Model
- title : String

Car
- registrationNum
- year : int
- licenseNumber : String
+ moveForward()
+ moveBackward()
+ stop()
+ turnRight()
+ turnLeft()

GearBoxType
- name : String
- remarks : String

GearBox
- gearRatio : float[]
- currentGear : int
+ shiftUp()
+ shiftDown()

Engine
- capacity : float
- numberOfCylinders : int
+ start()
+ brake()
+ accelerate()

Suspension
- springRate : float

Body
- numberOfDoors : int

Tire
- width : float
- airPressure : float

Wheel
- diameter : float

Brake
- type : String
+ apply()

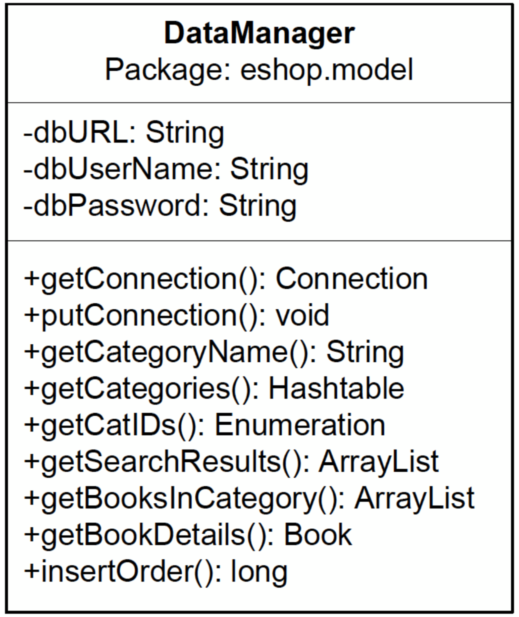
https://online.visual-paradigm.com/diagrams/templates/class-diagram/uml-class-diagram-example-car/

FACULDADE DE CIÊNCIAS E TECNOLOGIA

Introduction to UML

Class Diagrams:

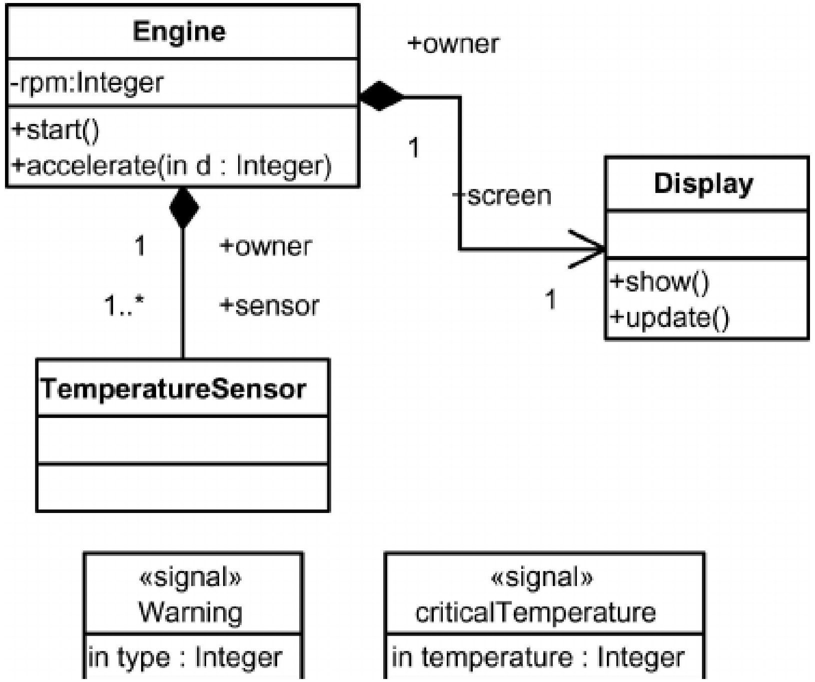
- Class name



DataManager
Package: eshop.model

-dbURL: String
-dbUserName: String
-dbPassword: String

+getConnection(): Connection
+putConnection(): void
+getCategoryName(): String
+getCategories(): Hashtable
+getCatIDs(): Enumeration
+getSearchResults(): ArrayList
+getBooksInCategory(): ArrayList
+getBookDetails(): Book
+insertOrder(): long

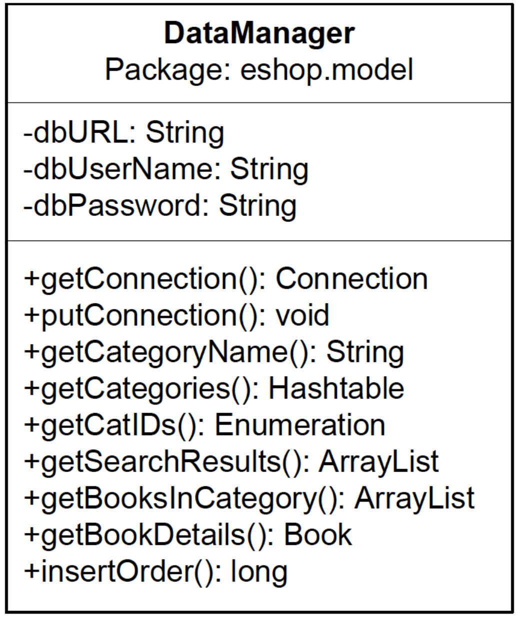


- Specific to objects
- meaningful to describe the aspect of the system

Introduction to UML

Class Diagrams:

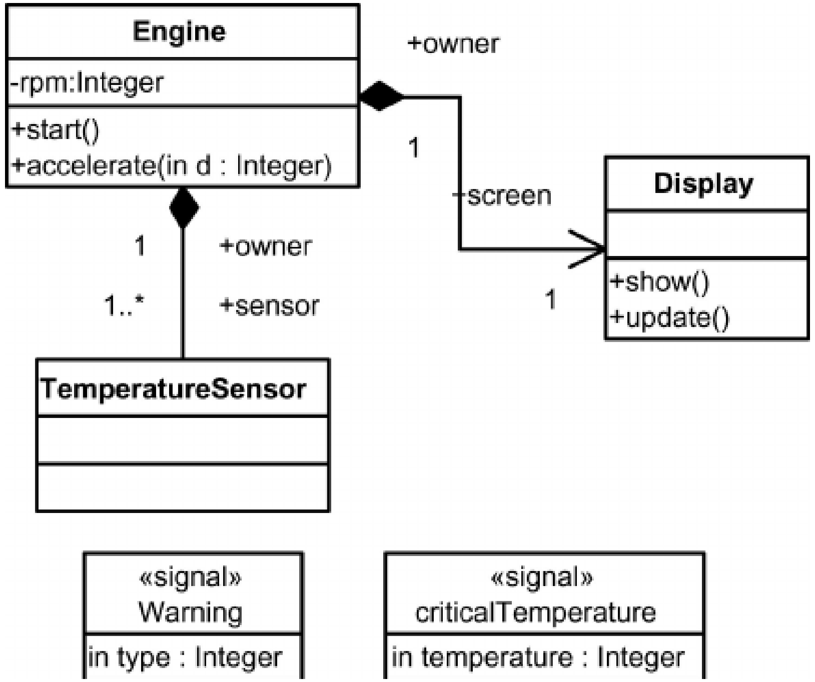
- Class attributes



DataManager
Package: eshop.model

-dbURL: String
-dbUserName: String
-dbPassword: String

+getConnection(): Connection
+putConnection(): void
+getCategoryName(): String
+getCategories(): Hashtable
+getCatIDs(): Enumeration
+getSearchResults(): ArrayList
+getBooksInCategory(): ArrayList
+getBookDetails(): Book
+insertOrder(): long

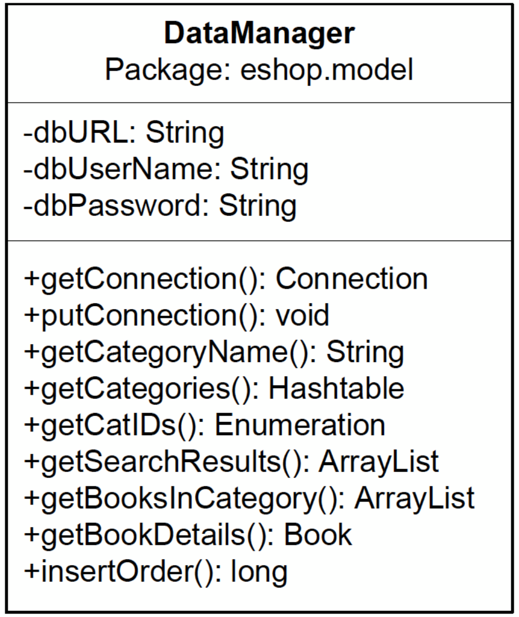


- Values that characterize the objects of a class
- Different types (eg integer, string, Boolean, date, …)

Introduction to UML

Class Diagrams:

- Class Methods / Operations

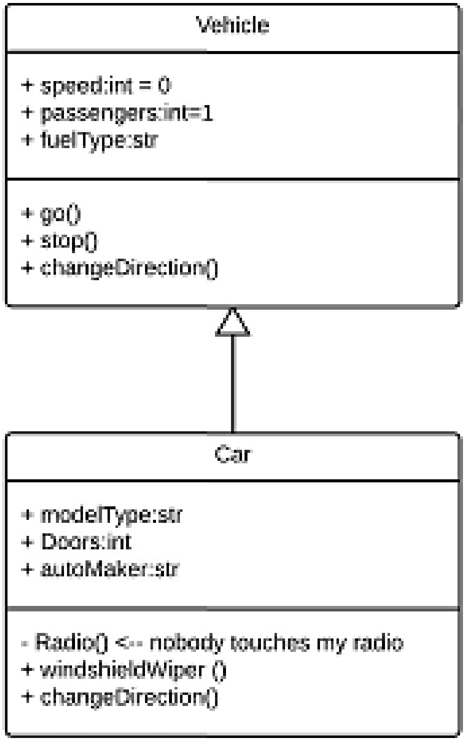


DataManager
Package: eshop.model

-dbURL: String
-dbUserName: String
-dbPassword: String

+getConnection(): Connection
+putConnection(): void
+getCategoryName(): String
+getCategories(): Hashtable
+getCatIDs(): Enumeration
+getSearchResults(): ArrayList
+getBooksInCategory(): ArrayList
+getBookDetails(): Book
+insertOrder(): long

  - Behaviours of the class



Vehicle

+ speed:int = 0
+ passengers:int=1
+ fuelType:str

+ go()
+ stop()
+ changeDirection()

Car

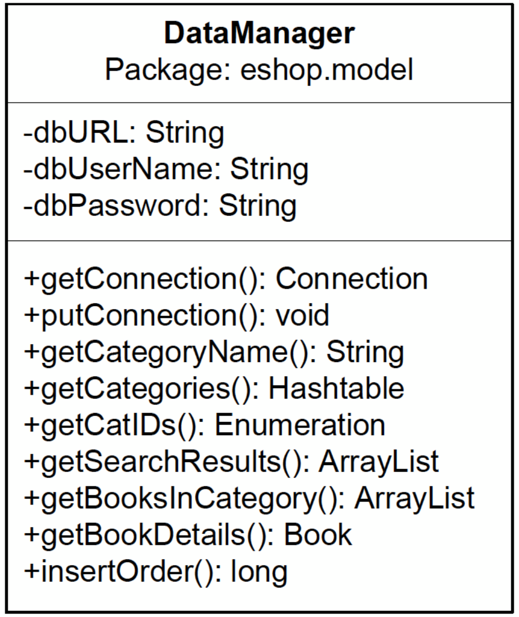
+ modelType:str
+ Doors:int
+ autoMaker:str

- Radio() <-- nobody touches my radio
+ windshieldWiper ()
+ changeDirection()

Introduction to UML

Class Diagrams:

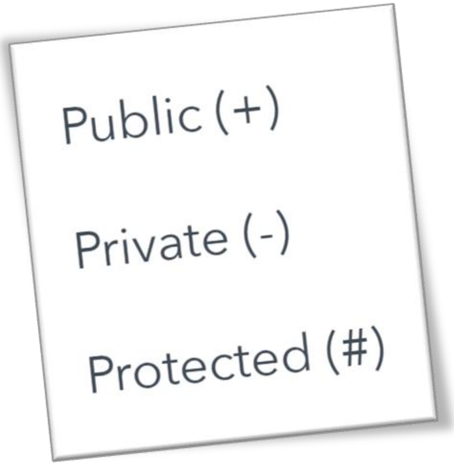
- Visibility of Methods / Operations



DataManager
Package: eshop.model

-dbURL: String
-dbUserName: String
-dbPassword: String

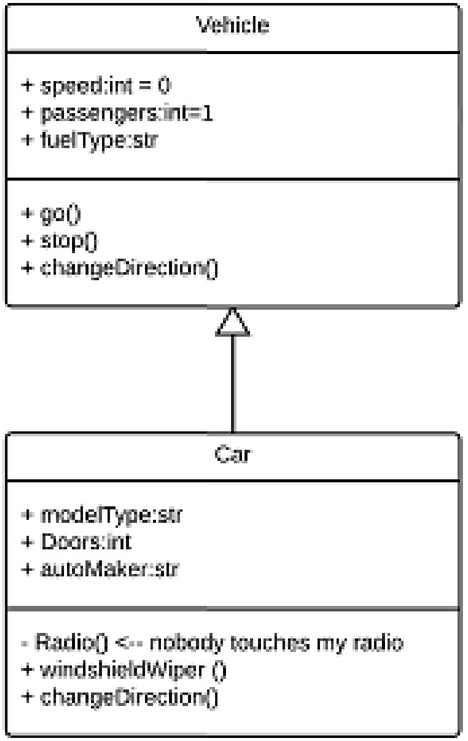
+getConnection(): Connection
+putConnection(): void
+getCategoryName(): String
+getCategories(): Hashtable
+getCatIDs(): Enumeration
+getSearchResults(): ArrayList
+getBooksInCategory(): ArrayList
+getBookDetails(): Book
+insertOrder(): long



Public (+)

Private (-)

Protected (#)



Vehicle

+ speed:int = 0
+ passengers:int=1
+ fuelType:str

+ go()
+ stop()
+ changeDirection()

Car

+ modelType:str
+ Doors:int
+ autoMaker:str

- Radio() <-- nobody touches my radio
+ windshieldWiper ()
+ changeDirection()

Introduction to UML

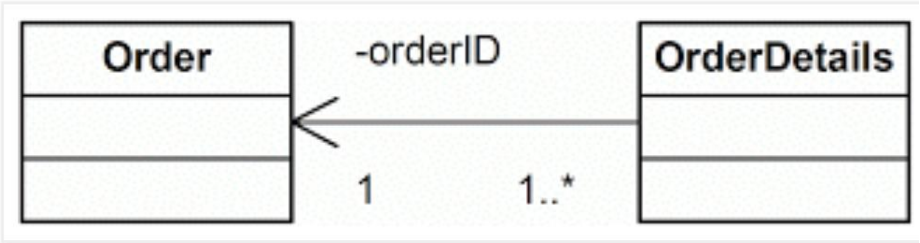
Class Diagrams:

- Relationships:
  - Relationships between classes can be of one of the following four types:
    - association,
    - generalization,
    - aggregation, and
    - composition

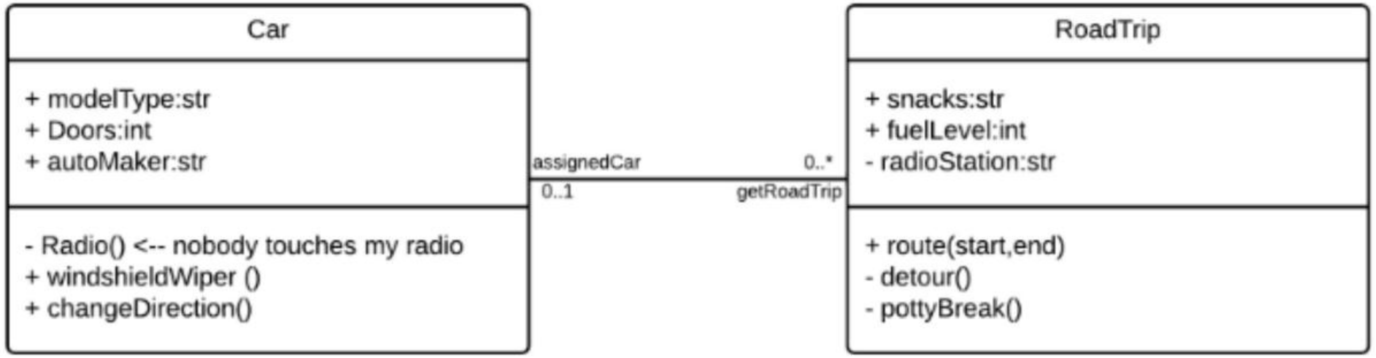
Introduction to UML

Class Diagrams:

- Associations:



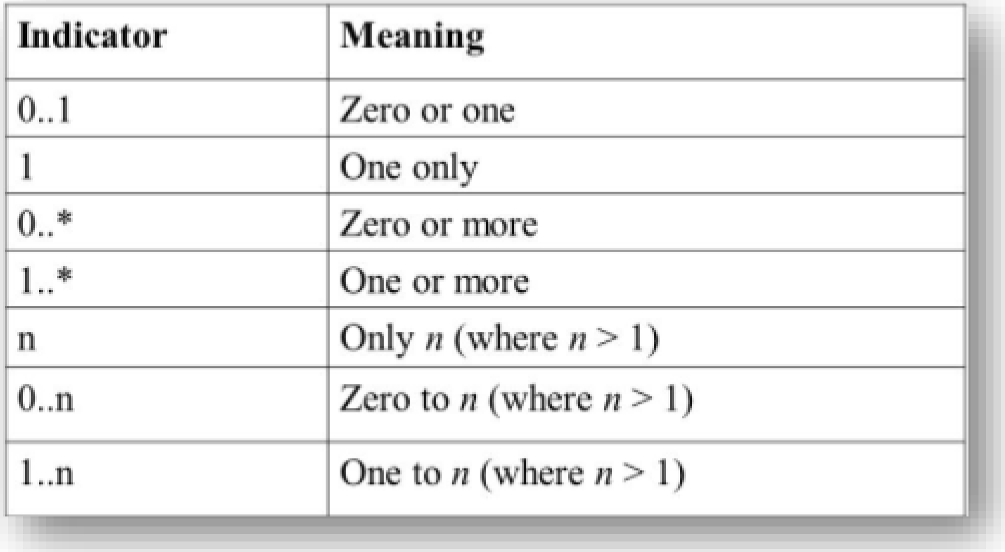
  - Associations model generic relationships between objects, possibly many to many.



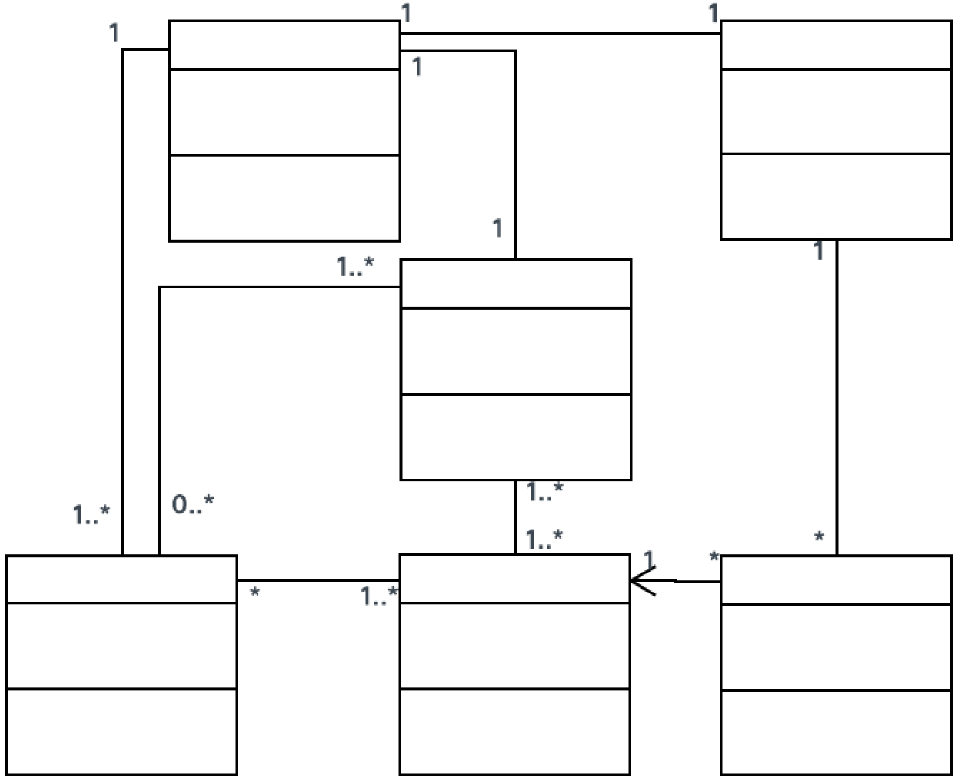
Introduction to UML

Class Diagrams:

- Multiplicity:



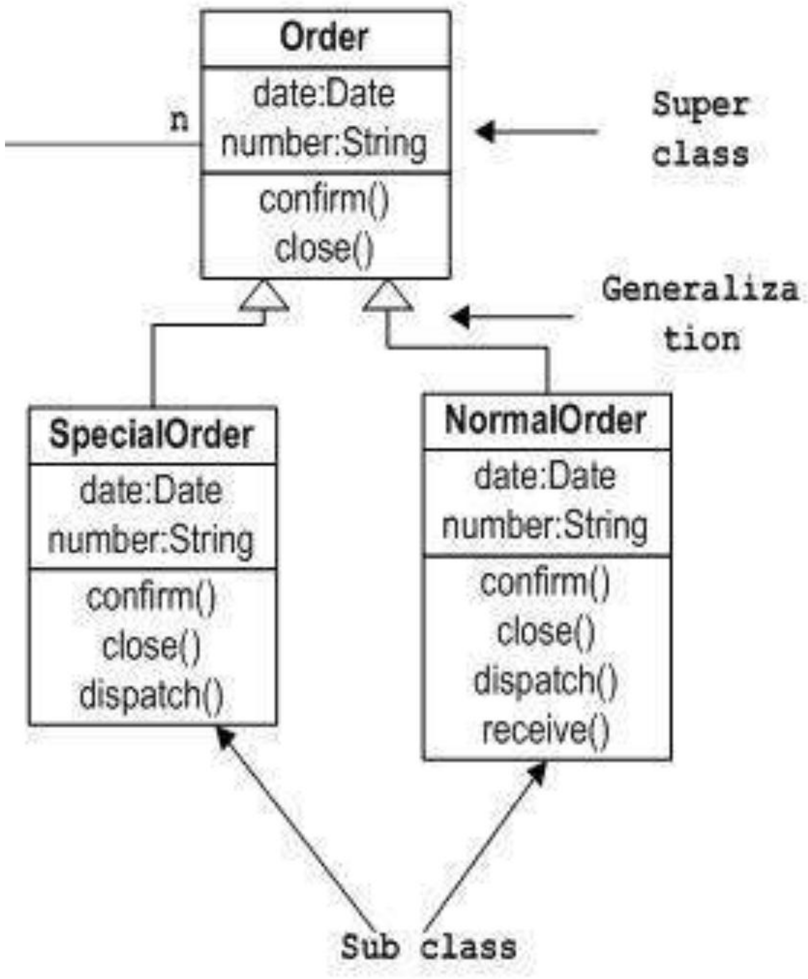
Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more
n	Only *n* (where *n* > 1)
0..n	Zero to *n* (where *n* > 1)
1..n	One to *n* (where *n* > 1)



Introduction to UML

Class Diagrams:

- Generalizations:
  - Generalizations model inheritance.
  - If class B extends class A, in UML it is said that A is a generalization of B.
  - A generalization is represented by an arrow with a white triangular head going from to the subclass to the superclass.

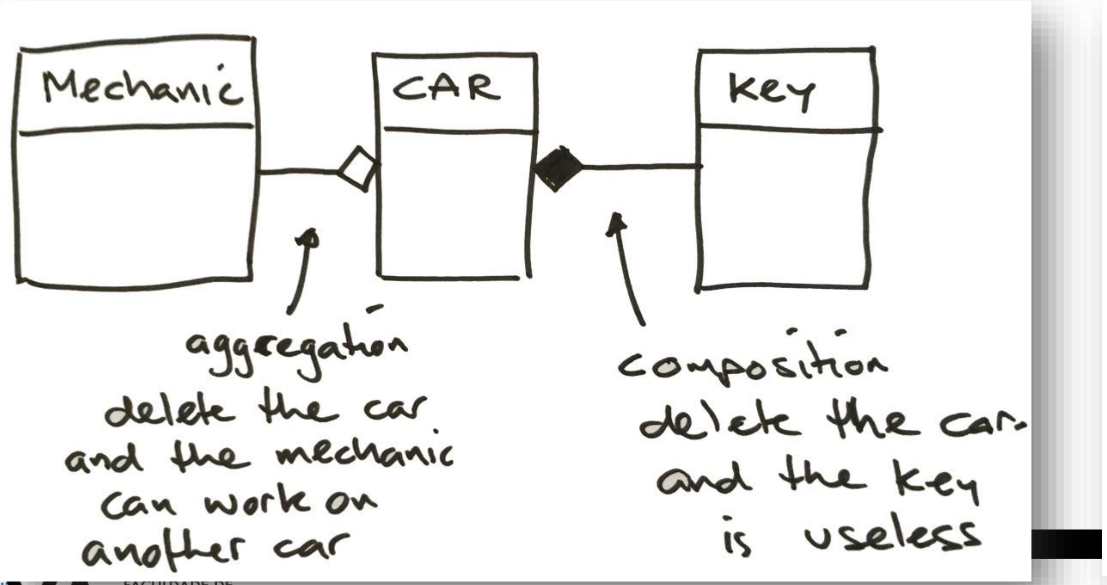


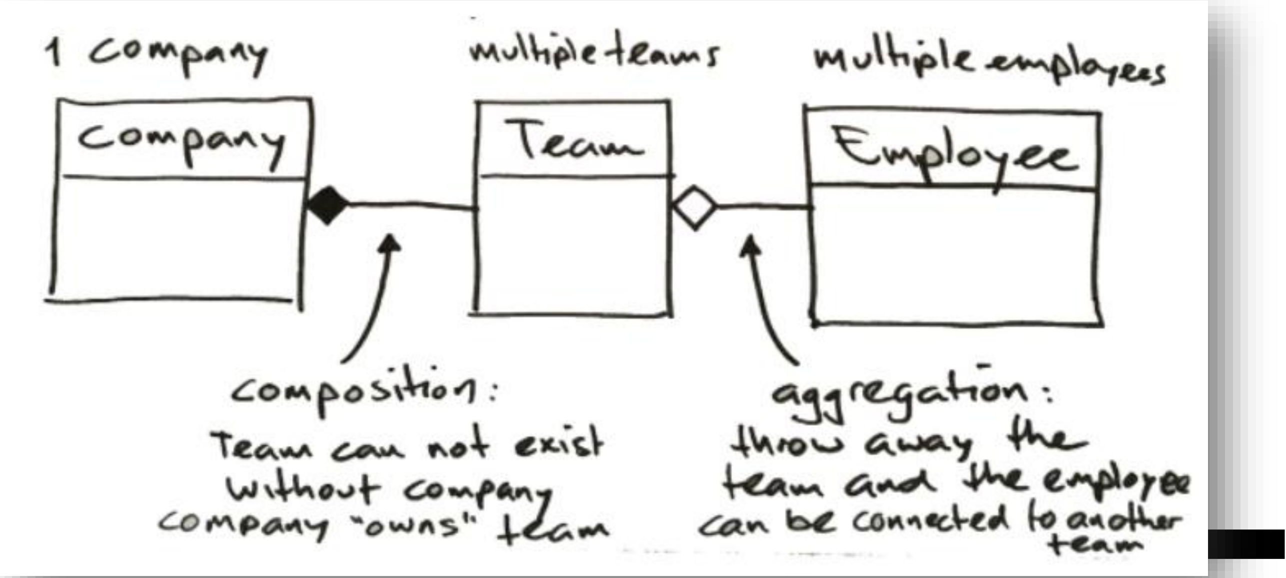
https://www.tutorialspoint.com/

Introduction to UML

Class Diagrams:

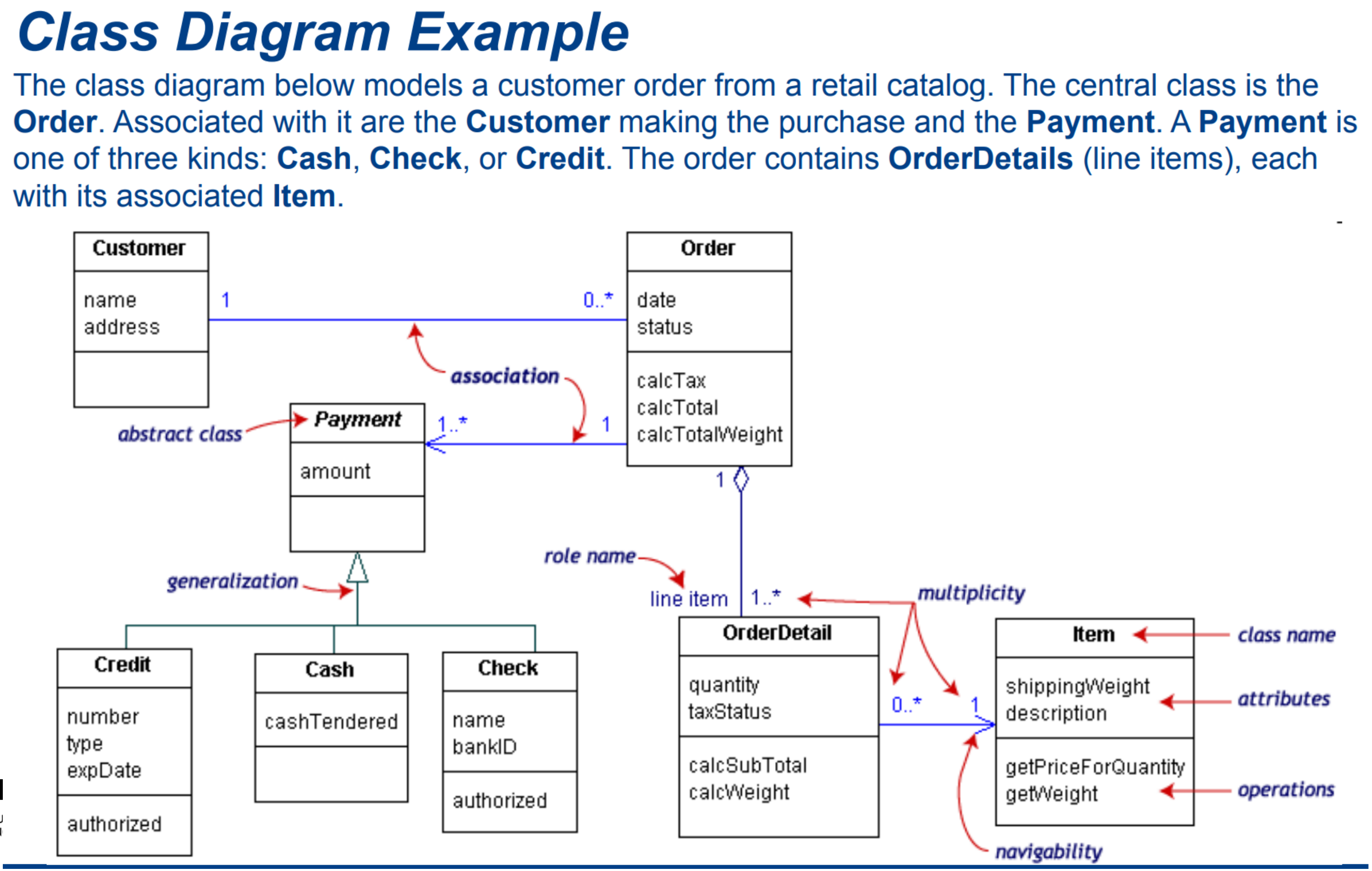
- Aggregations:
  - Aggregations model a special type of associations in which a "whole" is associated to its "parts".
- Compositions:
  - Compositions are like aggregations but indicate immutable containment. They model a form of containment in which container and contained objects belong together. If one ceases to exist so do the others.


Mechanic
CAR
Key
aggregation
delete the car and the mechanic can work on another car
composition
delete the car and the key is useless


1 company
multiple teams
multiple employees
Company
Team
Employee
composition:
Team can not exist without company
company "owns" team
aggregation:
throw away the team and the employee can be connected to another team

https://www.loekvandenouweland.com/content/difference-between-aggregation-and-composition.html

Introduction to UML

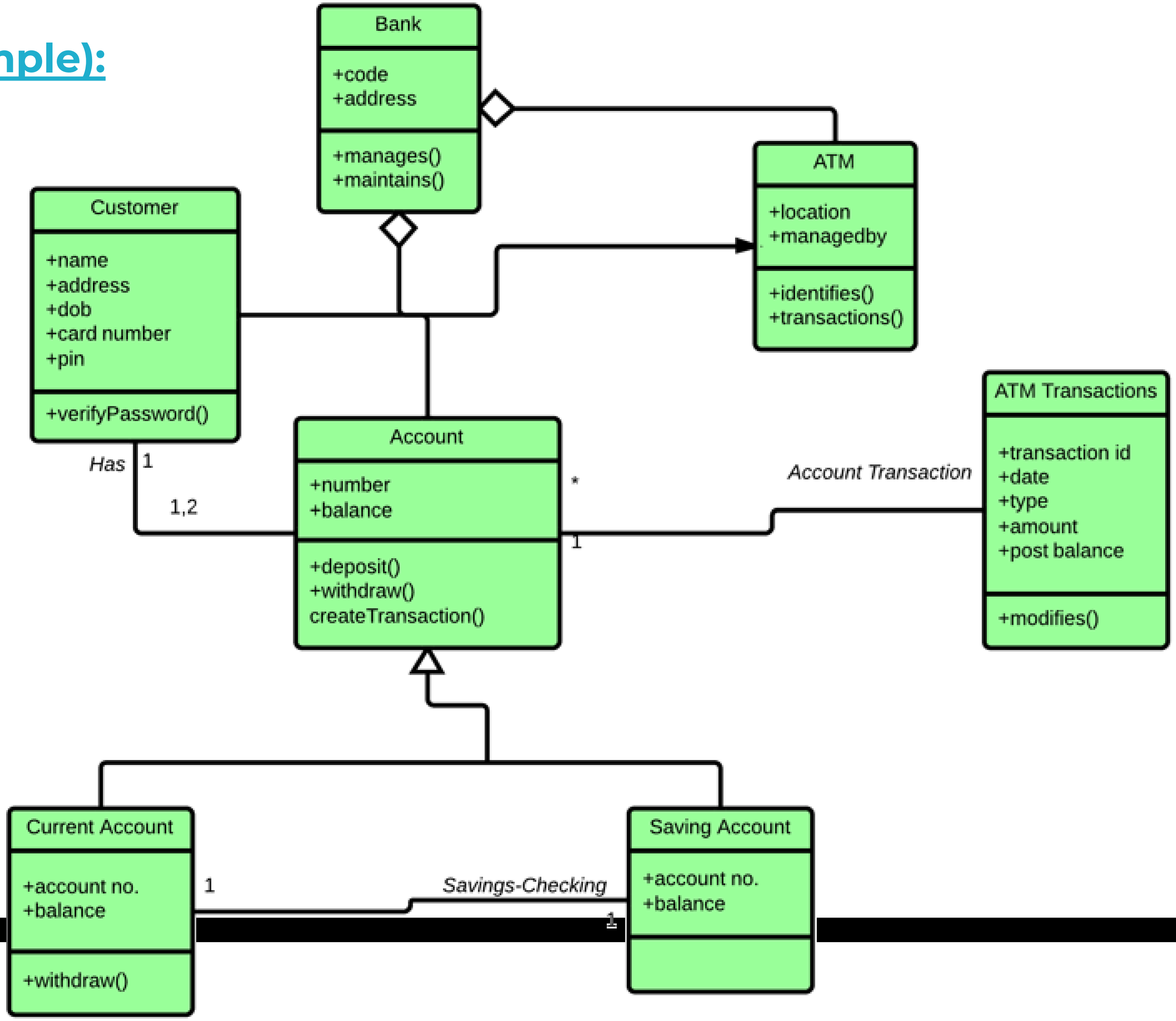


Class Diagram Example

The class diagram below models a customer order from a retail catalog. The central class is the **Order**. Associated with it are the **Customer** making the purchase and the **Payment**. A **Payment** is one of three kinds: **Cash**, **Check**, or **Credit**. The order contains **OrderDetails** (line items), each with its associated **Item**.

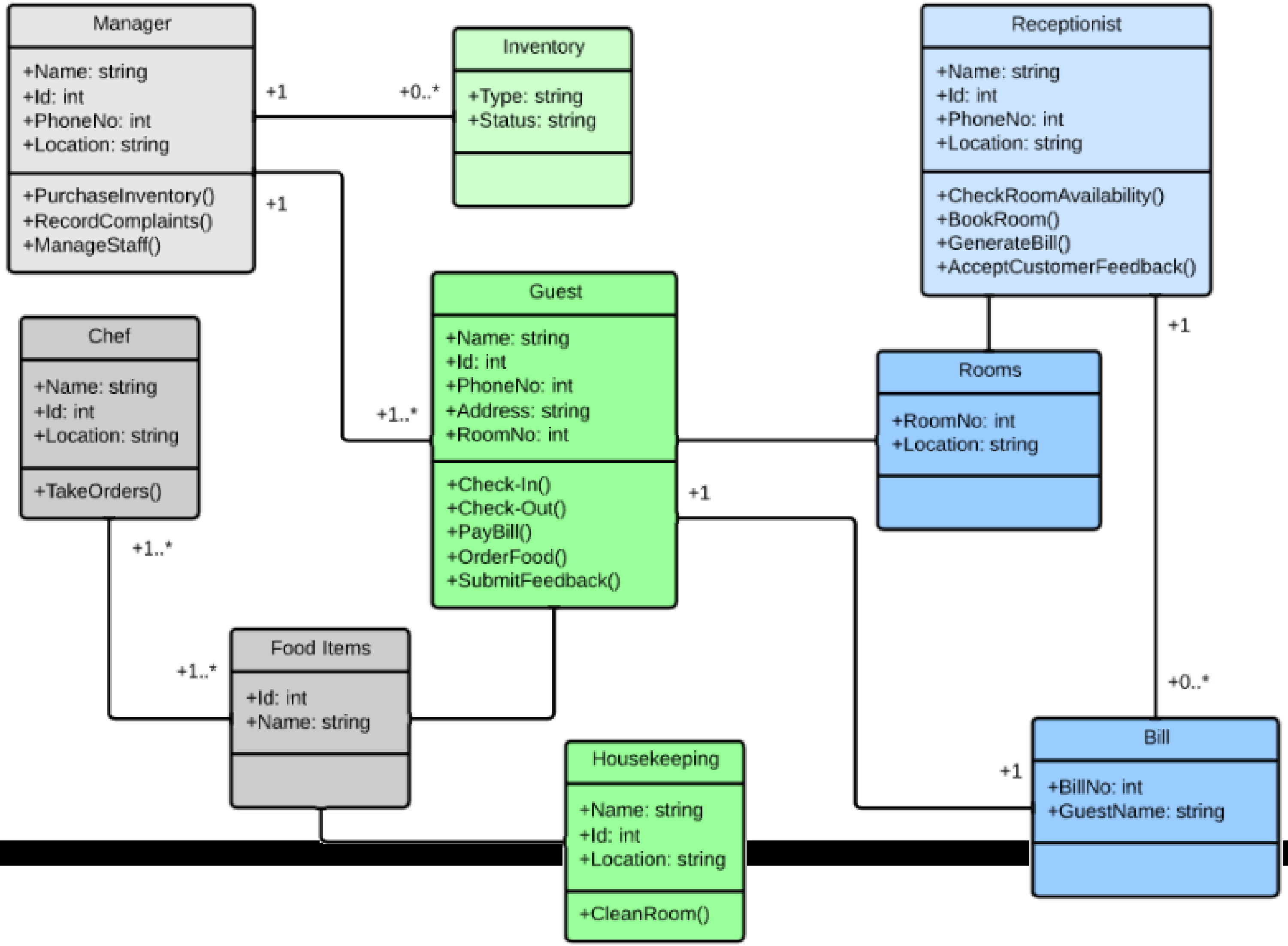
Introduction to UML

Class Diagrams (Example):



Introduction to UML

Class Diagrams (Example):



Manager
+Name: string
+Id: int
+PhoneNo: int
+Location: string
+PurchaseInventory()
+RecordComplaints()
+ManageStaff()

Inventory
+Type: string
+Status: string

Receptionist
+Name: string
+Id: int
+PhoneNo: int
+Location: string
+CheckRoomAvailability()
+BookRoom()
+GenerateBill()
+AcceptCustomerFeedback()

Guest
+Name: string
+Id: int
+PhoneNo: int
+Address: string
+RoomNo: int
+Check-In()
+Check-Out()
+PayBill()
+OrderFood()
+SubmitFeedback()

Chef
+Name: string
+Id: int
+Location: string
+TakeOrders()

Rooms
+RoomNo: int
+Location: string

Food Items
+Id: int
+Name: string

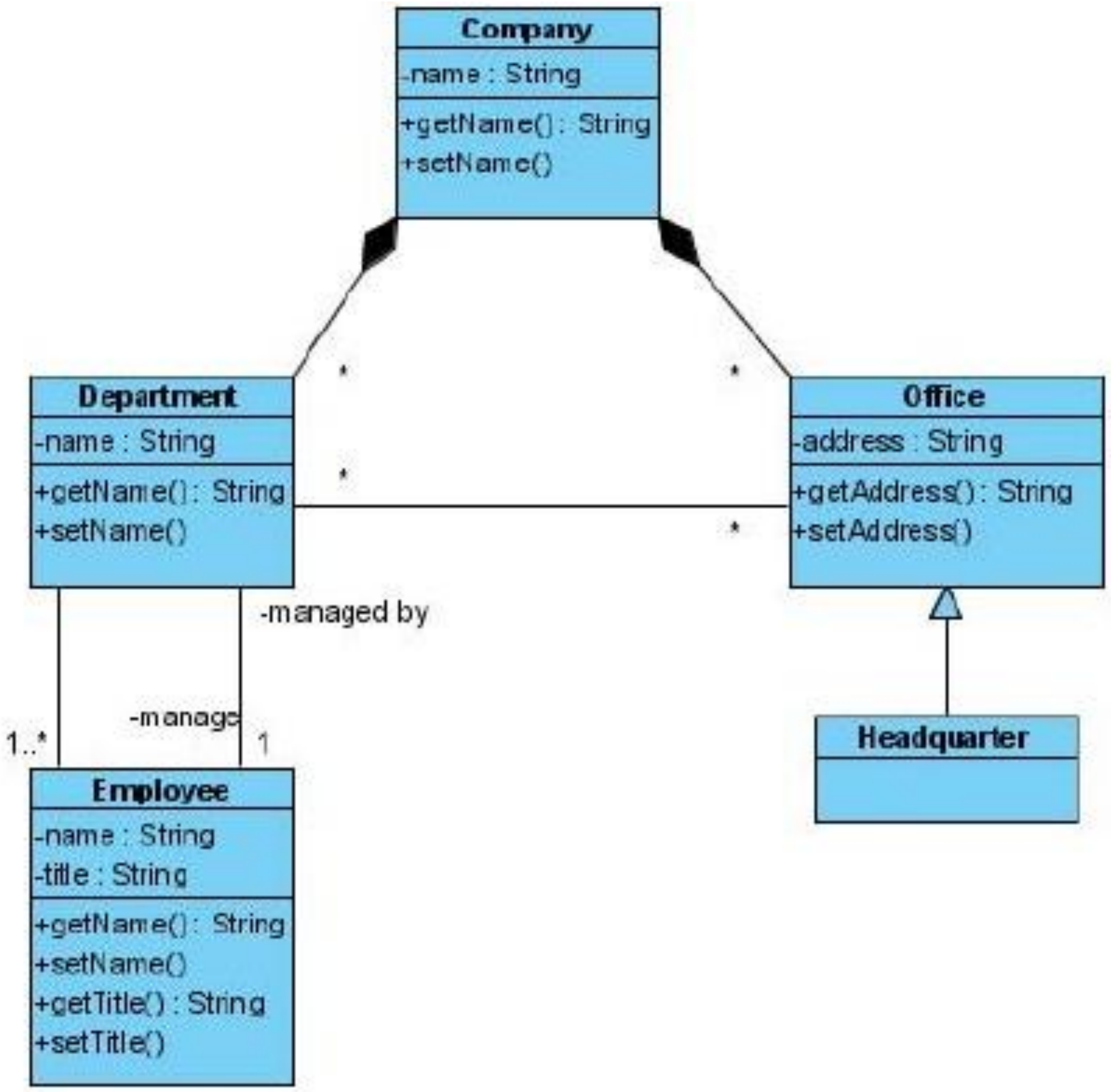
Housekeeping
+Name: string
+Id: int
+Location: string
+CleanRoom()

Bill
+BillNo: int
+GuestName: string

Introduction to UML

Class Diagrams (Exercise I):

- A company consists of departments.
- Departments are located in one or more offices.
- One office acts as a headquarter.
- Each department has a manager who is recruited from the set of employees.



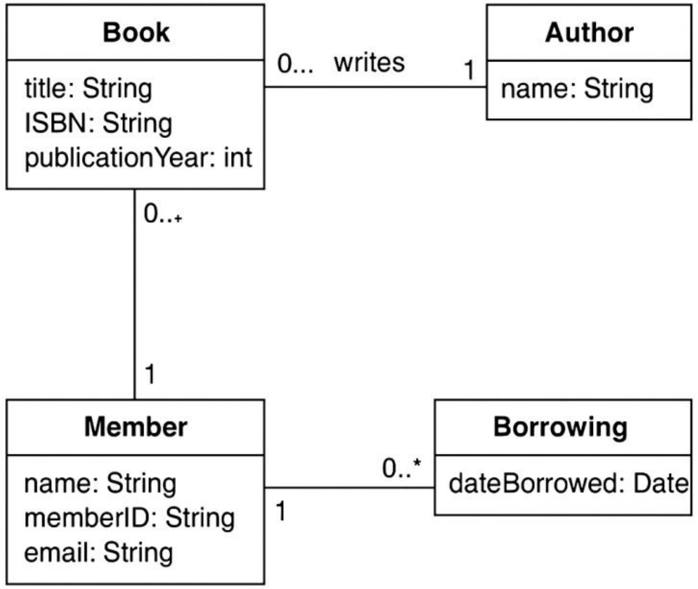
Introduction to UML

Class Diagrams (Exercise II):

The aim is to develop a system to support the management of **books and members in a Library**.

A local library wants to manage its books and members. The system should keep track of the following:

- The library has many Books. Each book has a title, ISBN, and publication year.
- Each Book has exactly one Author, but an Author can write multiple Books.
- The library has multiple Members. Each member has a name, member ID, and email address.
- A Member can borrow multiple Books, but a Book can be borrowed by only one Member at a time.
- Record the date borrowed for each borrowed book.



Book
title: String
ISBN: String
publicationYear: int

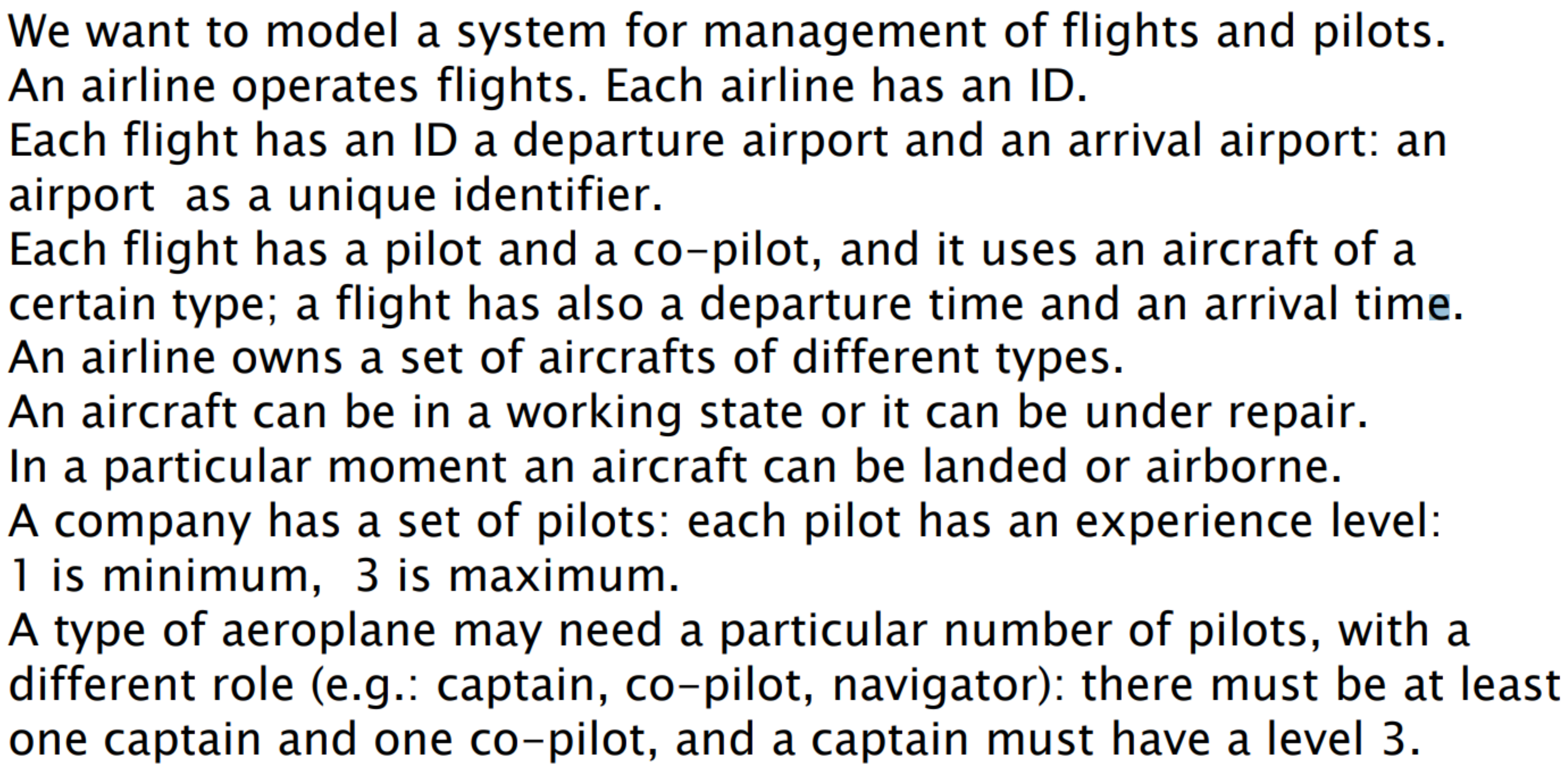
Author
name: String

Member
name: String
memberID: String
email: String

Borrowing
dateBorrowed: Date

Introduction to UML

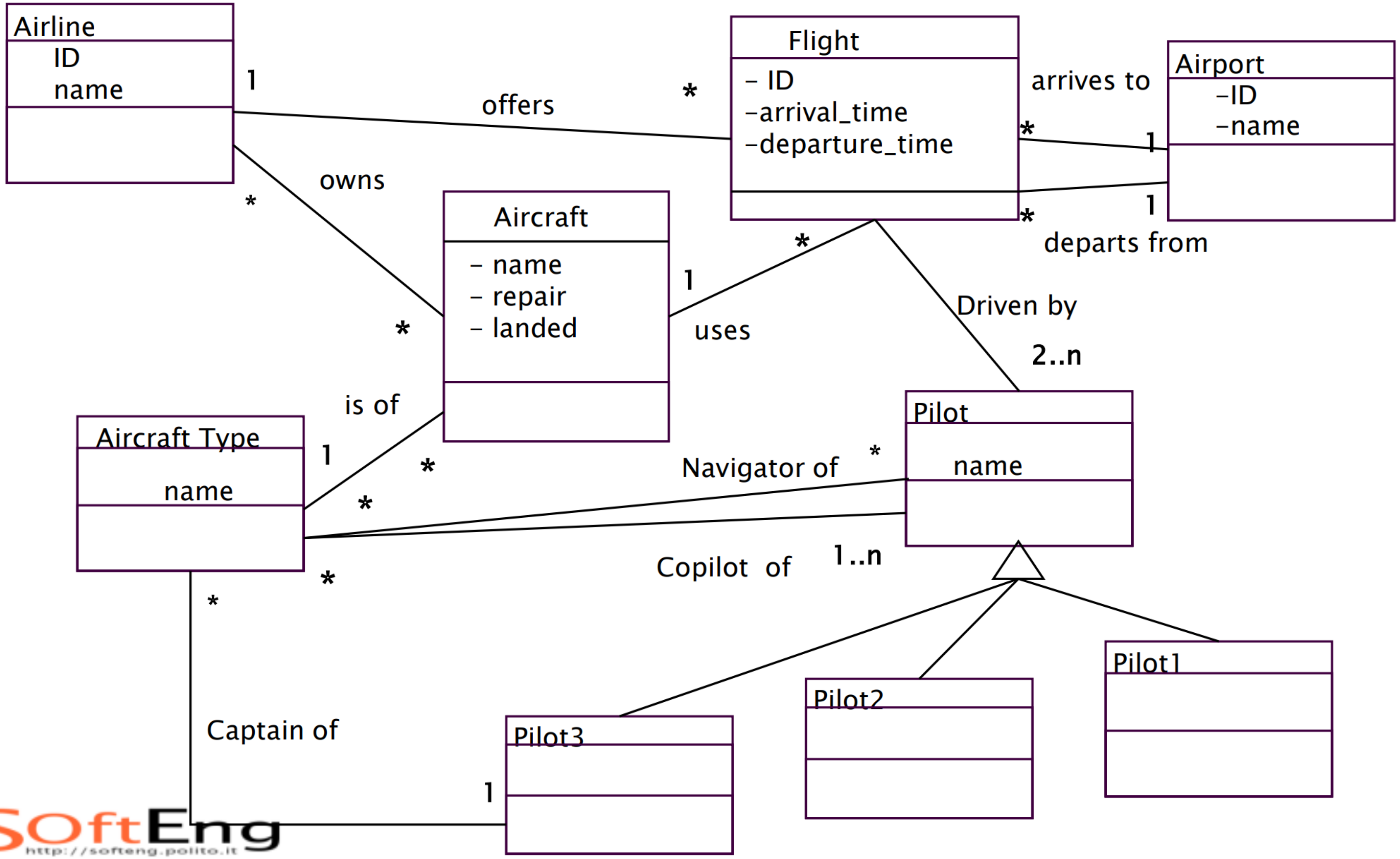
Class Diagrams (Exercise III):



We want to model a system for management of flights and pilots.
An airline operates flights. Each airline has an ID.
Each flight has an ID a departure airport and an arrival airport: an airport as a unique identifier.
Each flight has a pilot and a co-pilot, and it uses an aircraft of a certain type; a flight has also a departure time and an arrival time.
An airline owns a set of aircrafts of different types.
An aircraft can be in a working state or it can be under repair.
In a particular moment an aircraft can be landed or airborne.
A company has a set of pilots: each pilot has an experience level:
1 is minimum, 3 is maximum.
A type of aeroplane may need a particular number of pilots, with a different role (e.g.: captain, co-pilot, navigator): there must be at least one captain and one co-pilot, and a captain must have a level 3.

Introduction to UML

Class Diagrams (Exercise III):



Introduction to UML

Class Diagrams (Exercise IV):

The aim is to develop a system to support the management of **online magazines**.

The system will be used by editors, illustrators and photographers, who enter their content directly into the system. All these collaborators have a login and password.
Each magazine has a title and consists of several issues. Each issue has a date and year. Each issue is made up of articles. The article has a title, and is made up of boxes, which have a position in the issue, indicated by pages, coordinates and size. The boxes are images or text.
The images are done by illustrators or photographers, while the texts are written by editors. All of them are collaborators, and their names and initials must be kept.
A magazine also has a director that is an editor.

Further reading

- OMG – Unified Modeling Language. (http://www.uml.org/what-is-uml.htm)
- Hamilton, k., Miles, R., 2006, Learning UML 2.0: A Pragmatic Introduction to UML, O'Reilly
- Nunes, M., O’Neill, H., 2004, Fundamental de UML, FCA – Editora de Informatica
- Jackson, M., 1995, Software Requirements and Specification, Addison-Wesley
- Silva, A., Videira, C., (2001), UML Metodologias e Ferramentas, Centro Atlântico
- Models and Modelling, The Open University, 2016 (http://www.open.edu/)
- Visual-Paradigm, What is Data Flow Diagram?(https://www.visual-paradigm.com)
- Business analysis excellence. (http://business-analysis-excellence.com)
- UML Diagram Tutorial for Beginners (https://www.guru99.com/uml-tutorial.html)
- UML Tutorial (https://www.javatpoint.com/uml)

https://drsmithbiology.weebly.com/further-reading.html

Good Work!

Ana Inês Oliveira

NOVA School of Sciences and Technology | FCT NOVA

aio@fct.unl.pt