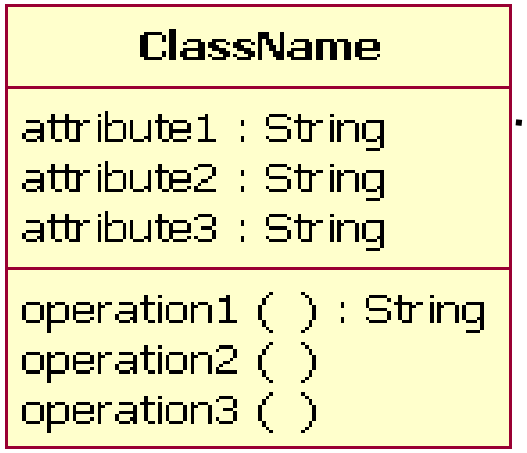
UML class diagrams – what is a CLASS

- Class:
  - A class is a description of objects.
  - All objects of a class share the same attribute description, associations to classes, operations and methods.
  - A class represents a concept in a model.
  - A class does have a name, a set of attributes, a set of operations and constraints

UML Class Diagrams – Elements of classes

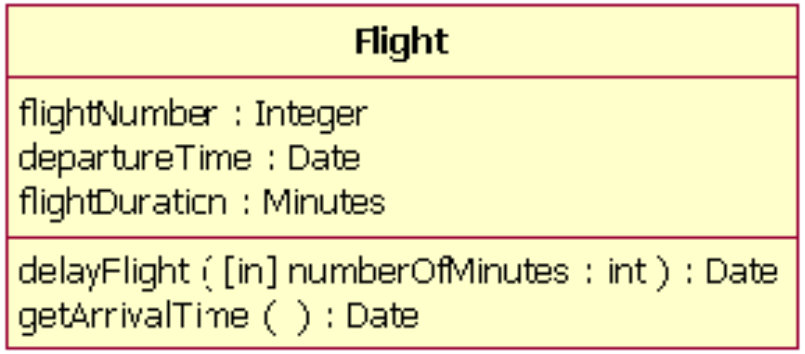


Name of the class (Singular)

Attributes of the Class

Operations (or behavior) of the Class (Methods/ Functions)

An example:



UML Class Diagrams – class attributes

- The attribute section of a class lists each of the class's attributes on a separate line. The attribute section is optional, but when used it contains each attribute of the class displayed in a list format
- UML notation for an attribute has the form:

  <<stereotype>> [visibility] name [multiplicity] [:type] [= initial value] [{property-string}]

  Example:

  + origin [0..1] : Point

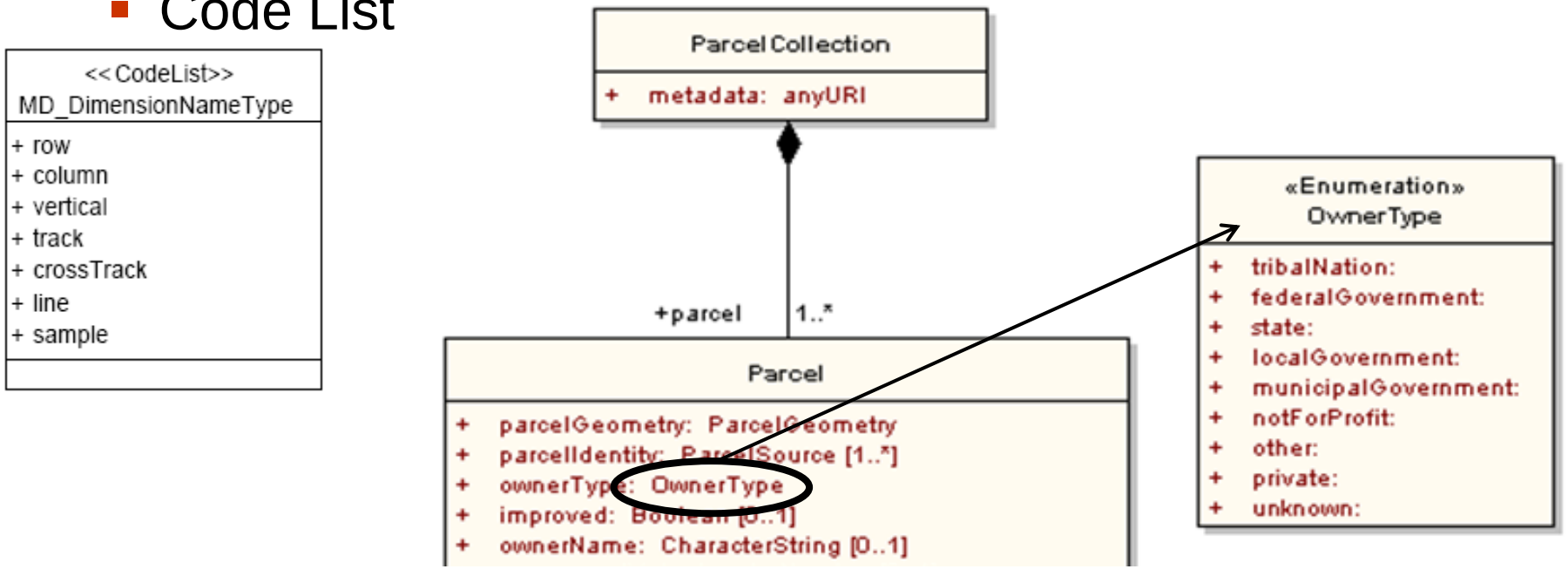
  (Note: multiplicity 0..1 means that this attribute is optional. If no explicit multiplicity is given, it is assumed to be 1.)
- All attributes ***must be typed*** (i.e. data type must be defined) and the type must exist among the set of legal base types, the constructed/defined types. A type must always be specified, there is no default type.

UML Class Diagrams – Data Types

- In ISO 19103, Data Types are grouped into three categories
  - Primitive types: Fundamental types for representing values, examples are CharacterString, Integer, Boolean, Date, Time, etc.
  - Implementation types: Template types for representing multiple occurrences of other types, examples are Set, Bag, Sequence and Record.
  - Derived types: Measure types and units of measurement.

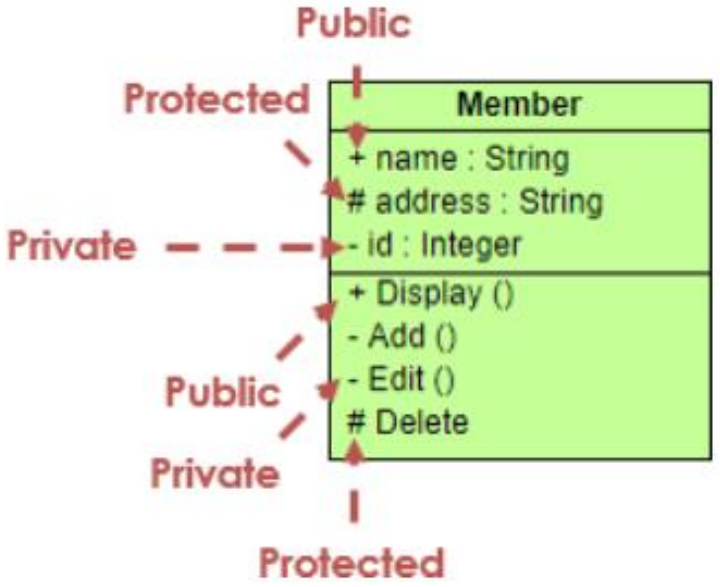
UML Class Diagrams – data types

- Example:
  - Code List



Visibility

Sign	Visibility Type
+	public
#	protected
-	Private
~	package



Access Right	public (+)	private (-)	protected (#)	Package (~)
Members of the same class	yes	yes	yes	yes
Members of derived classes	yes	no	yes	yes
Members of any other class	yes	no	no	in same package

Visibility continued (additional background)

Note, that if a named element is not owned by any namespace, then it does not have a visibility (in UML namepaces are grouped by the means of “packages” that contains elements). A **namespace** in UML is a container for named element.

- A *public element* is visible to all elements that can access the contents of the namespace that owns it. Public visibility is represented by '+' literal.
- A *package element* is owned by a namespace that is not a package, and is visible to elements that are in the same package as its owning namespace. Only named elements that are not owned by packages can be marked as having package visibility. Any element marked as having package visibility is visible to all elements within the nearest enclosing package (given that other owning elements have proper visibility). Outside the nearest enclosing package, an element marked as having package visibility is not visible. Package visibility is represented by '~' literal.
- A *protected element* is visible to elements that have a generalization relationship to the namespace that owns it. Protected visibility is represented by '#' literal.
- A *private element* is only visible inside the namespace that owns it. Private visibility is represented by '-' literal.

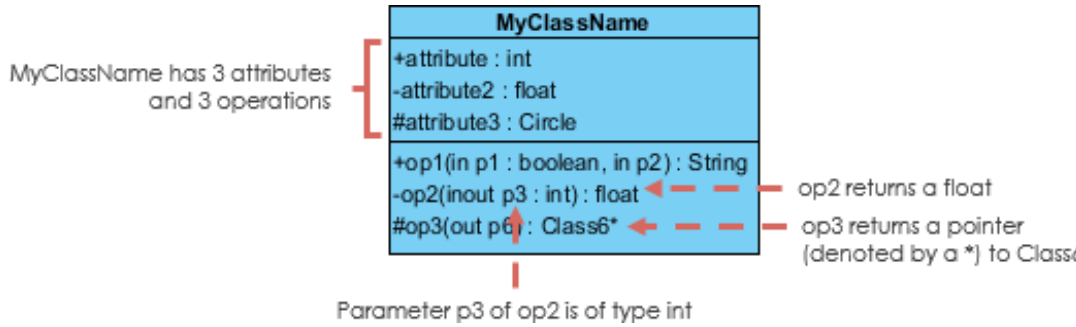
(taken from: https://www.uml-diagrams.org/visibility.html, last visited Dec-2022)

UML Class diagrams – class operations

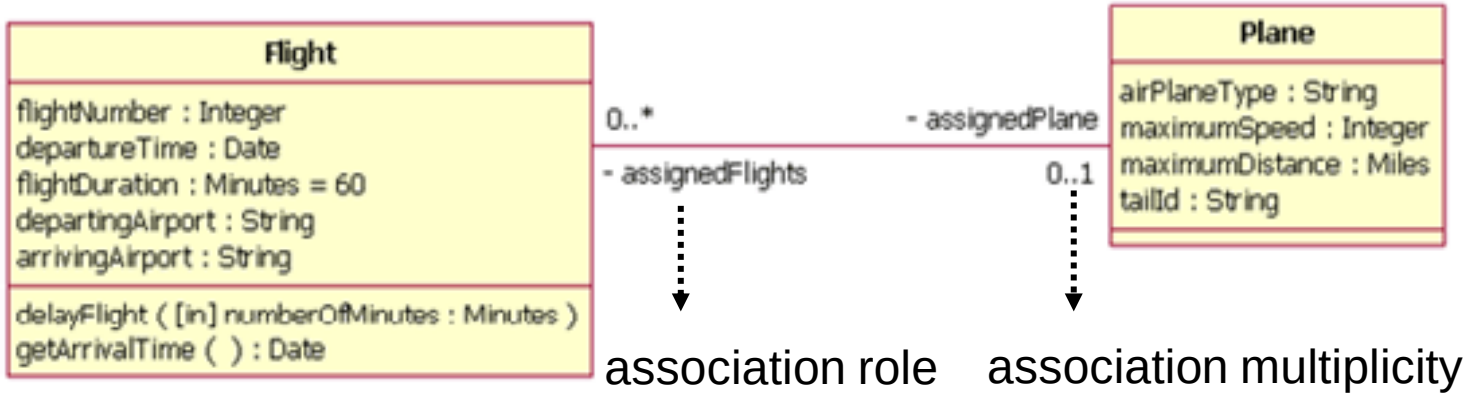
- The operations are documented in the bottom compartment of the class diagram's rectangle, which also is optional. Like the attributes, the operations of a class are displayed in a list format, with each operation on its own line
- UML notation for operations:

  [visibility] name [(parameter-list)] [:return-type] [{property-string}]

  - [(parameter-list element)] ::=
    - [direction] name : type [= default-value]
    - [direction] ::= in | out | inout („in“ is the „default“ Value)
- Example:
  - + getBalance (day: date=01/01/2021): Currency



Association (Bi - directional)

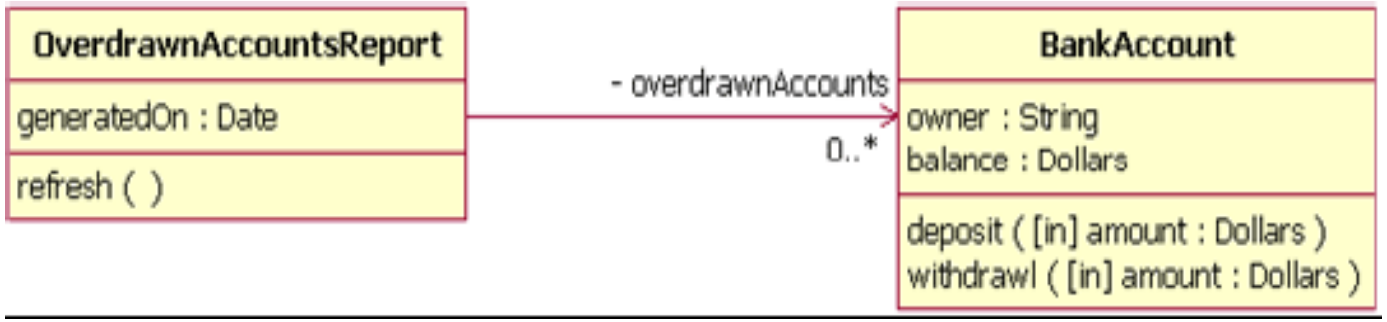


Associations describe the connection between classes. In bi-directional associations, both classes are aware that they are in a participation.
Bi-directional associations are the most commonly used association – therefore frequently referred to as “standard association”

Association (Multiplicity)

Indicator	Meaning
0..1	Zero or one
1	Exactly one (no more and no less)
0..*	Zero or many
1..*	One or many
n	Only *n* (where *n* > 1)
0..n	Zero or *n* (where *n* > 1)
1..n	One to *n* (where *n* > 1)

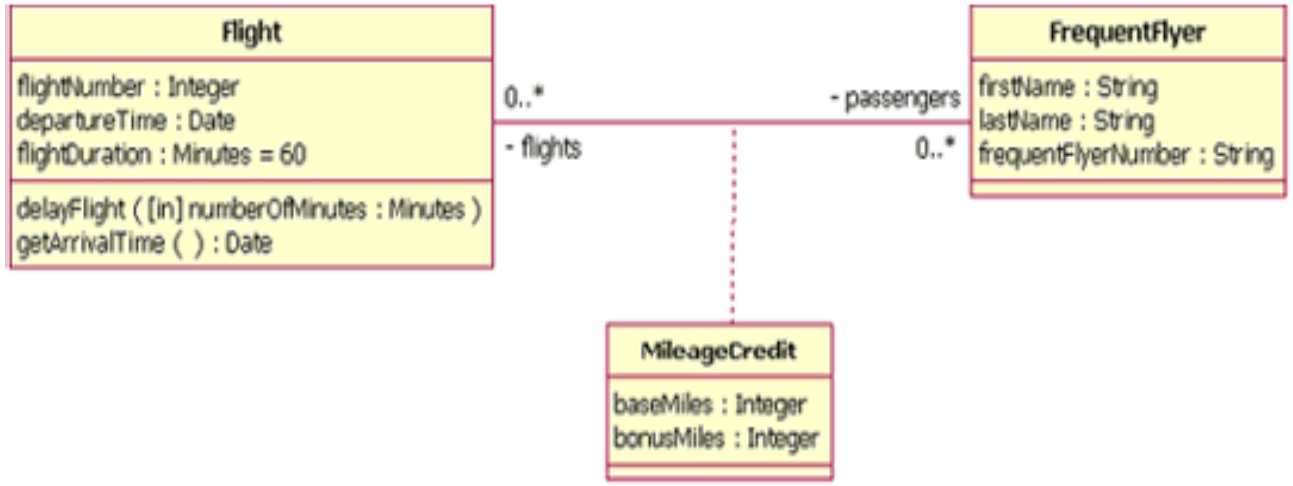
Associationen (Uni - directional)



Like Bi-directional association, but the connected class (class, where the arrow is pointing to) does not know about the existence of the other class.

In the example:
„OverdrawnAccountsReport“ do know about the existence of „BankAccount, but „BankAccount“ does not know about „OverdrawnAccountsReport“.
„BankAccount“ can play the role of „overdrawnAccounts“.
OverdrawnAccountsReport is a “reporting class” to report on the business class “BankAccount”

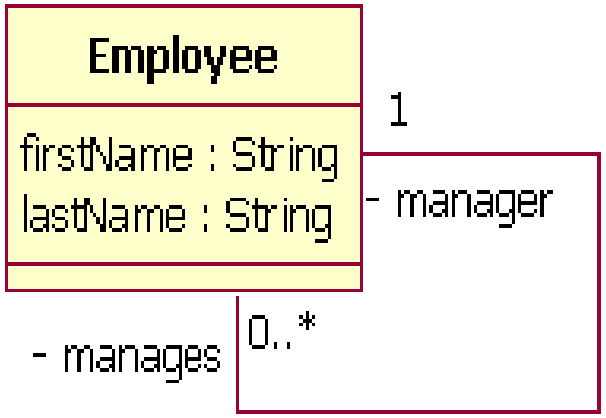
Association – Association Classes



Specifies the association between 2 classes

Example: The association between the class „Flight“ and „FrequentFlyer“ does require the class „MileageCredit“ (this means, that if a frequent fliers is booking a legitimate flight to earn miles it has to be accounted to his MileageCredit)

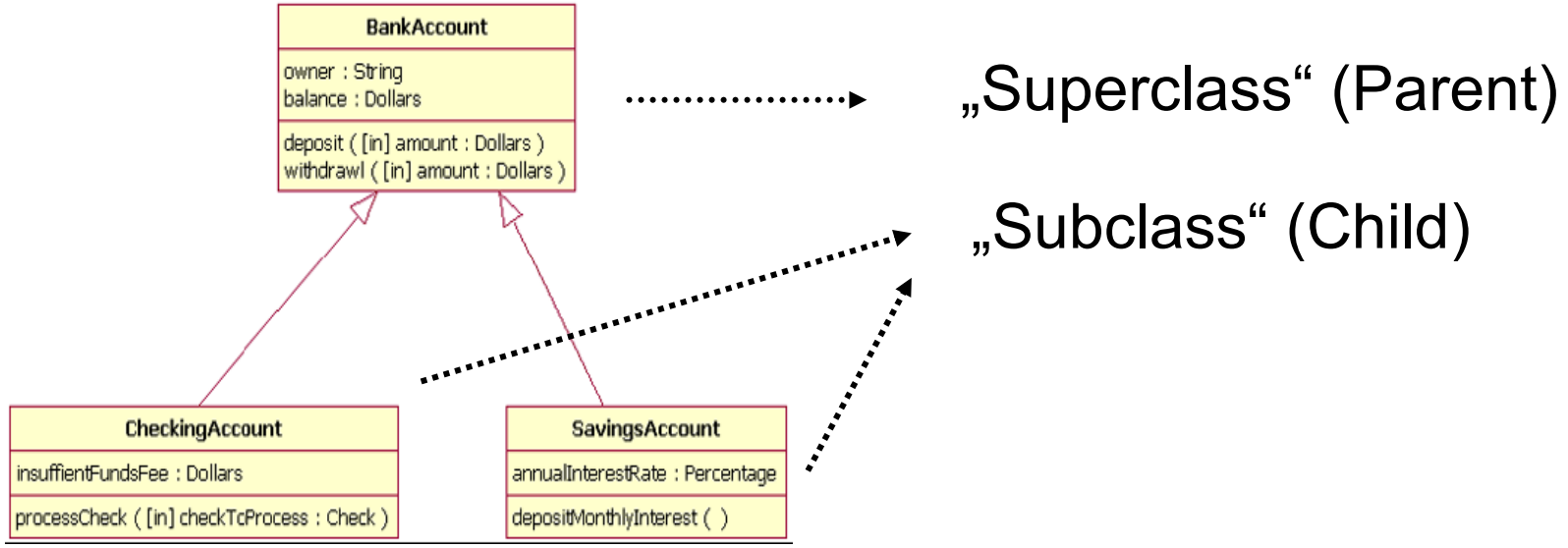
Recursive (reflexive) – association



Class do have an association with itself.

Example:
An employee of a class „employee“ can hat the role „manger“ or can have the role „be managed“

Inheritance / Generalization



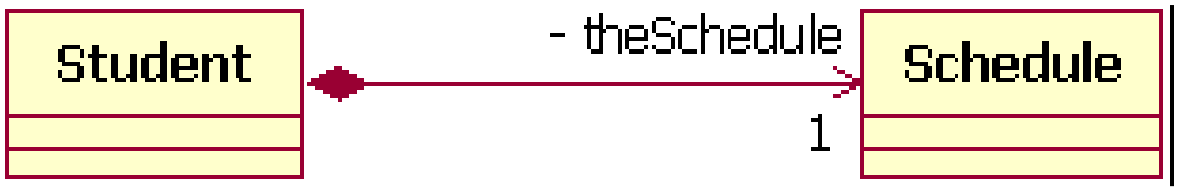
Subclasses *inherits all operations and properties of the parent class*, but may contain additional operations and properties

Note: instead of inheritance frequently the term „generalization“ is used – sometimes also “kind of relationship”

Aggregation – Base-Aggregation

A class is part of another class.
Example:
„Department“ is part of a „Company“. A „Company“ is constituted of minimal one or more „departments“.
Also called: „*Part of relationships*“.

Aggregation – Composite-Aggregation



Like Base Aggregation. A class is part of another class, but the existence of the dependent class is dependent on the existence of the aggregation class.

Example:
The class „Schedule“ is part of the class „Student“. The „Schedule“ class cannot exist without the „Student“ class. the This is called a „*Strong Aggregation*“.

When are class diagrams used?s

1. Describing the static view of the system.
2. Modeling the collaboration among the elements of the static view.
3. Describing the functionalities performed by the system.
4. Construction of software applications using object oriented languages.
5. Performing code forward engineering for the target systems
6. Classifying classes or components as library for future reuses

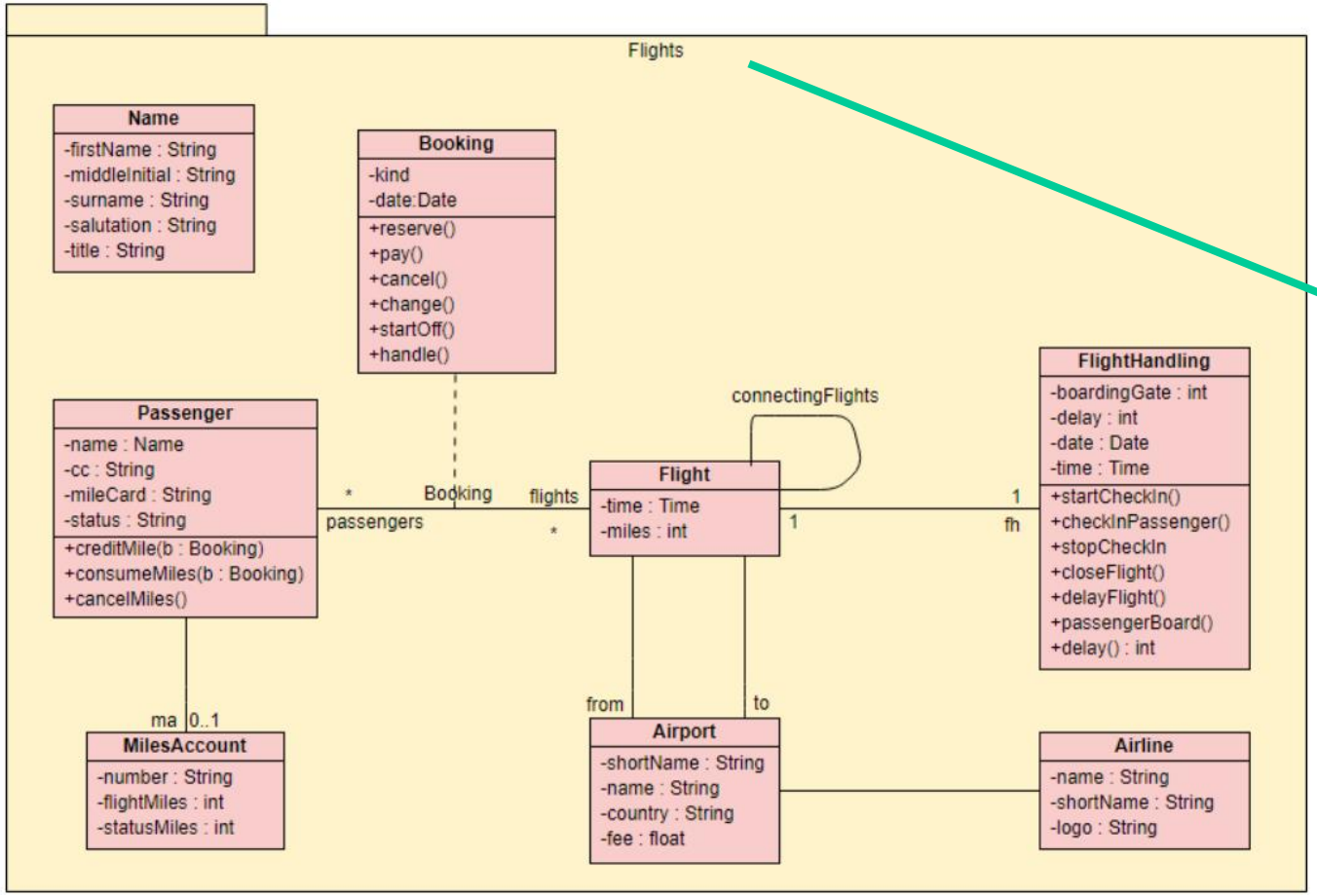
(taken from: https://online.visual-paradigm.com/diagrams/tutorials/class-diagram-tutorial/ , last visited Dec-2022)

How to draw class diagrams

1. Identify the objects in the problem domain, and create classes for each of them. (e.g. Teacher, Student, Course for an enrollment system)
2. Add attributes for those classes (e.g. name, address, telephone for the Student class)
3. Add operations for those classes (e.g. addStudent(student) for the Course class)
4. Connect the classes with appropriate relationships (e.g. Relate Teacher and Course with an association)
5. Optionally specify the multiplicities for association connectors' ends (e.g. Input 0..3 for the Course side of the association that connects Teacher and Course, to signify that one teacher can teach multiple up to three courses)

(taken from: https://online.visual-paradigm.com/diagrams/tutorials/class-diagram-tutorial/ , last visited Dec-2022)

Class Diagram example



packages to categorize classes (logical segmentation of classes in packages)

(taken from: https://online.visual-paradigm.com/diagrams/tutorials/class-diagram-tutorial/ , last visited Nov-2021)

Use of Spatial Standards in UML

- UML is an accredited Conceptual Schema Language used by all major standardization organizations für spatial information
- It defines principles / guidelines and framework of the use of UML for spatial data modelling
  - Class attribute definitions (spatial and non-spatial)
  - Class operations (spatial and non-spatial)
  - Relationships (spatial and non-spatial)
- Uses a specific terminology (defined by standards)

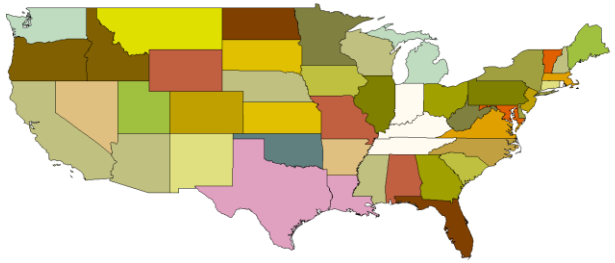
Terminology according to ISO TC211

- A ***feature*** is an abstraction of a real world phenomenon.
- A feature is geographic *if it is associated with a location relative to the Earth.*
- A digital representation of the real world can be thought of as a set of features ***(feature type***) or as a single instance ***(feature instance)***
- The term features is frequently used to describe discrete data entities whose position in space is described by geometric and topological primitives such as points, lines or polygons. However - ISO TC211 includes ***coverages*** by the term feature.

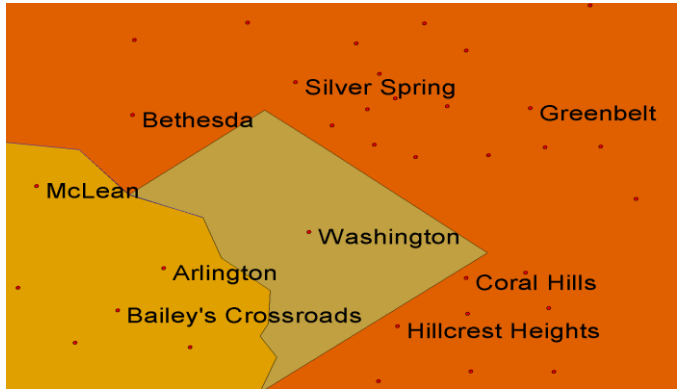
Terminology according to ISO TC211

- **Feature Attributes**
  - Describes the characteristics of a feature
  - Consists of a Name, Type and Value
  - E.g. A feature attribute named ‘length’ may have an attribute value ’82.4’ which belongs to the data type ‘real’.
- **Feature Operation**
  - operation that every instance of a feature type may perform
  - An operation upon the feature type 'dam' is to raise the dam. The result of this operation is to raise the level of water in a reservoir.
- **Schema**
  - formal description of a model
- **„universe of discourse“**
  - view of the real or hypothetical world that includes everything of interest

Features: Examples



Name: John Doe
Employee Id: 342
Hiring Date: 02/02/02
Title: Senior Computer Engineer
Extension: 1126
Address: 7855 Walker Drive, Greenbelt MD



Feature Instances, Types and Collections

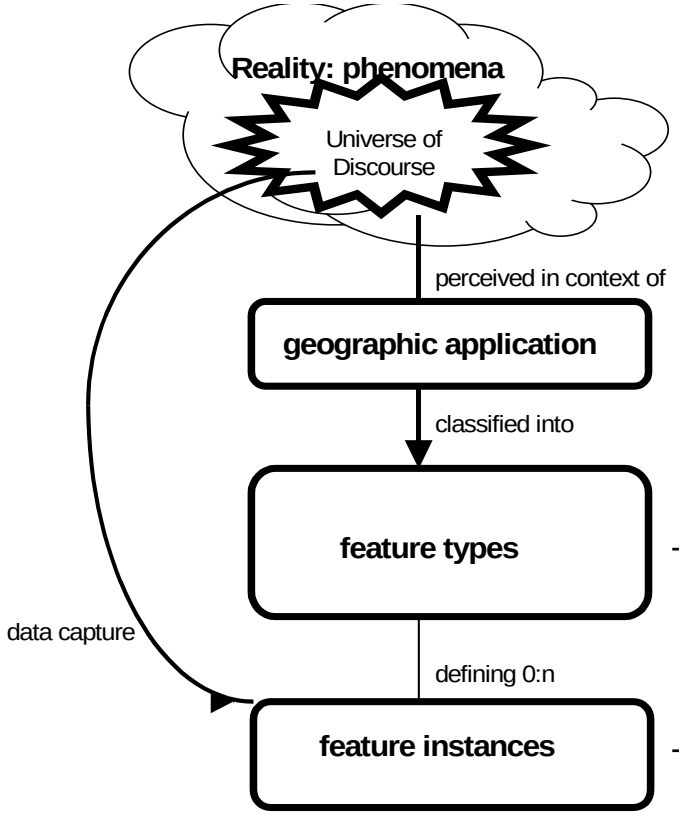
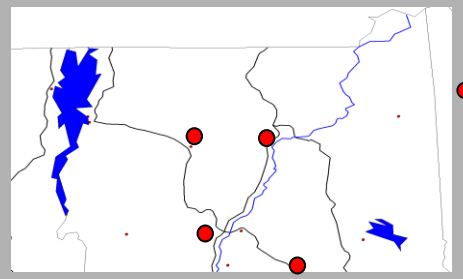


Figure © ISO

Feature collection
Collection of features that can itself be regarded as a feature (it has its own feature type and distinct properties in addition to the features it contains)

Examples: feature collection City containing other features like roads, rivers, colleges, etc.



→ **Type:** groups feature instances into classes with common characteristics*

Examples: Highway, Tower

→ **Instance:** represented as discrete phenomenon associated with its geographic and temporal coordinates

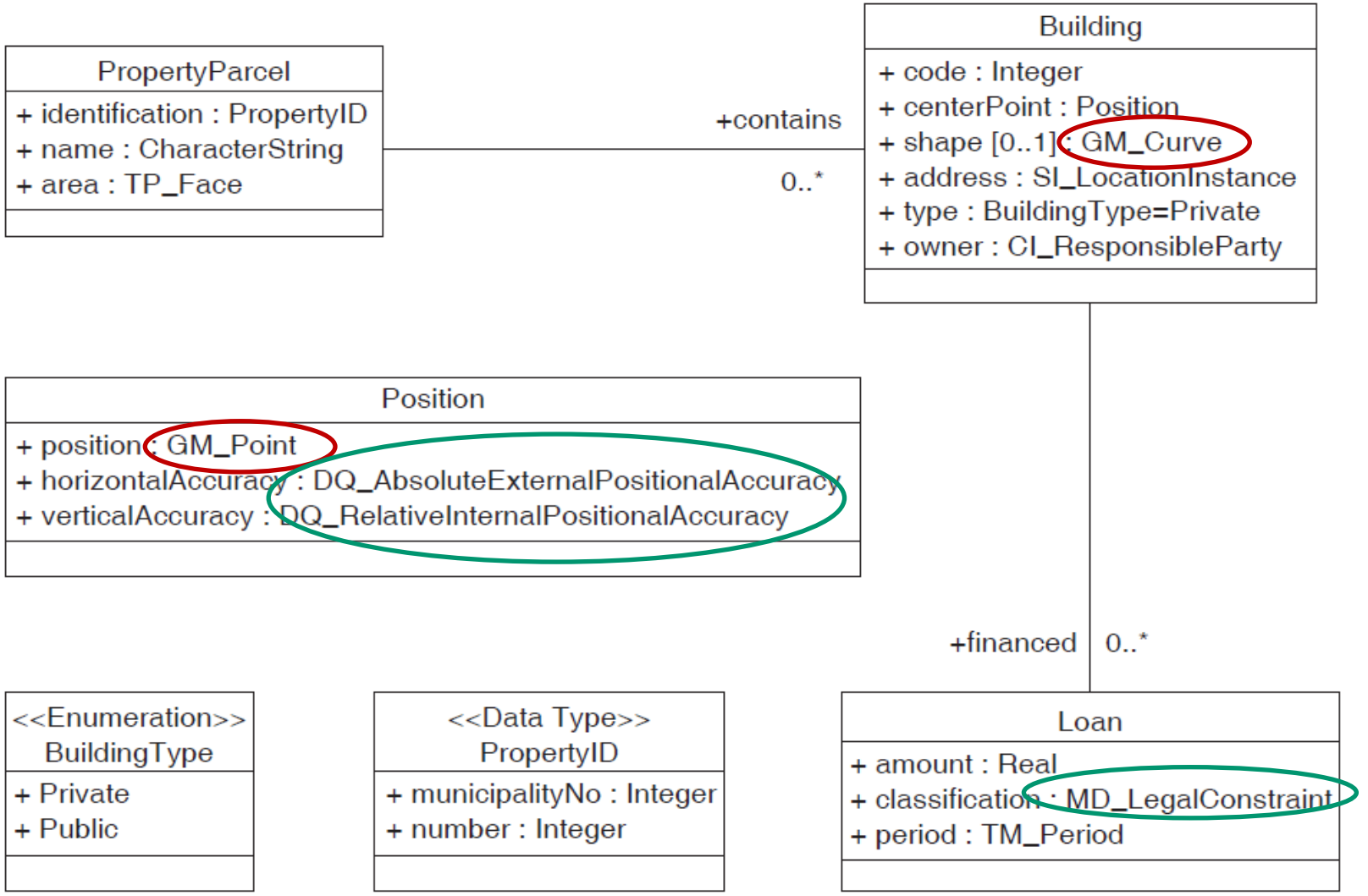
Examples: Interstate 95, Eiffel Tower

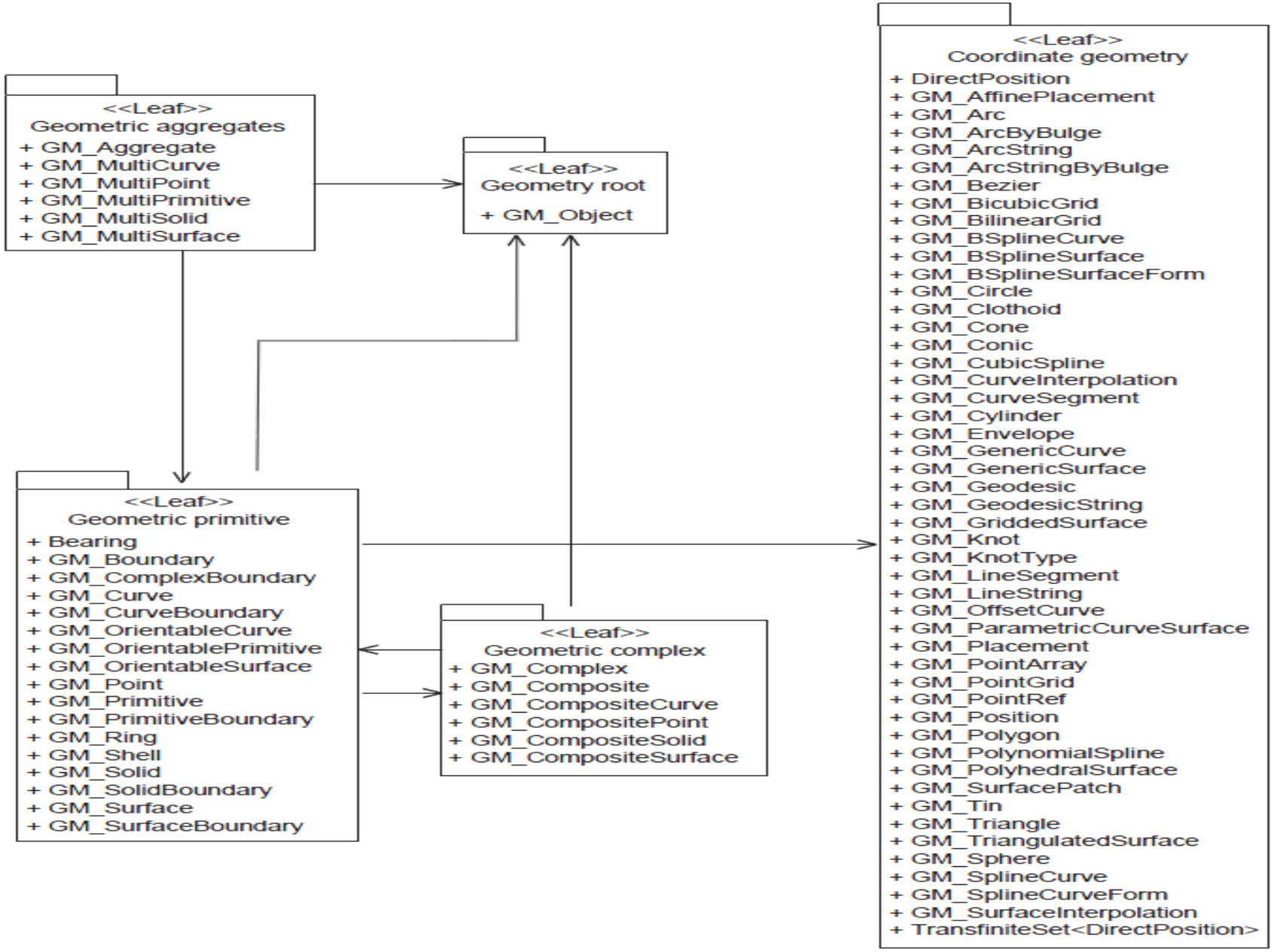
ISO 19109 Rules for application schema specifies how data shall be organized to reflect the particular needs of applications with similar data requirements.

Feature and UML Classes

- **In UML features are modeled as UML classes**
  - Feature attributes
  - Feature operations
- **However, (spatial) Features are special**
  - Spatial Attribute
    - Geometries
    - Location (implicit by address or other references)
  - Spatial operations
    - Spatial analysis functions

Example of a spatial information model – in UML class diagram notation





Geometry classes according to ISO 19107

ISO Geometry definitions (ISO 19125 – Simple Feature)
END