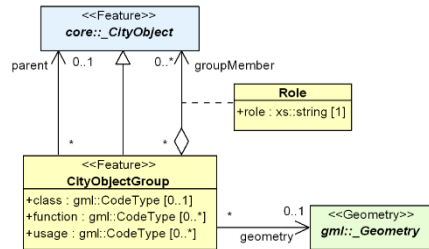


UML class diagrams in a nutshell



Giorgio Agugiaro

Last update: 31 December 2024



License

This presentation is licensed under the [Creative Commons License CC BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/). According to CC BY-NC-SA 4.0 permission is granted to share this document, i.e. copy and redistribute the material in any medium or format, and to adapt it, i.e. remix, transform, and build upon the material under the following conditions:



- **Attribution:** You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **NonCommercial:** You may not use the material for commercial purposes.
- **ShareAlike:** If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- **No additional restrictions:** You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

UML in a nutshell

UML stands for ***Unified Modelling Language*** and is a general-purpose, development modelling language in the field of software engineering that is intended to provide a standard way to visualise the design of a system.



When working with **class diagrams**, three are the main items to look for:

- **Classes** ("what?")
 - Attributes
 - Methods (not treated in these slides)
- **Multiplicity** ("how many?")
- **Relations** ("how do classes relate to each other?")

UML in a nutshell

Intro

Classes

Relations

Packages

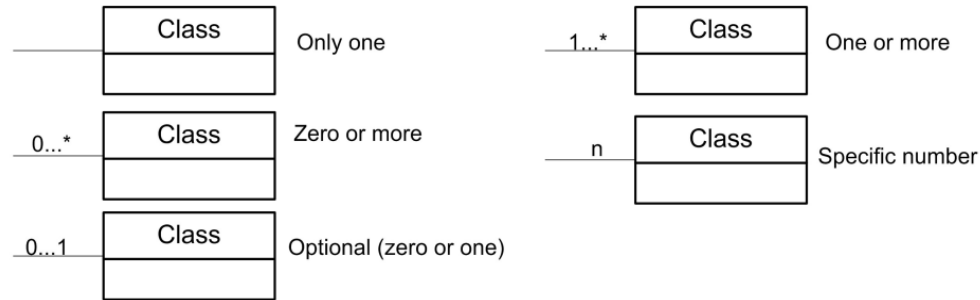
Namespaces

Association between classes

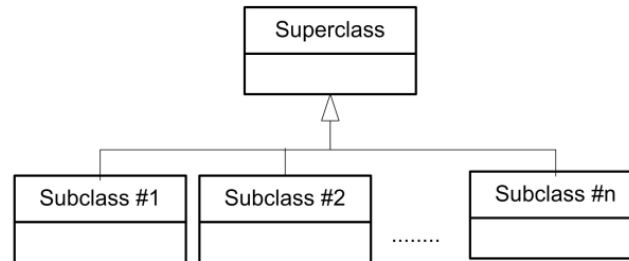


This is a simple way of reading such a relation, e.g. "Class 1 **has** Class 2". Alternatively, it is explicitly indicated by the "Role" word, always written near the target.

Association multiplicity



Class inheritance (subtyping)



"is a (superclass of)"

"is a (subclass of)"

UML in a nutshell

Intro

Classes

Relations

Packages

Namespaces

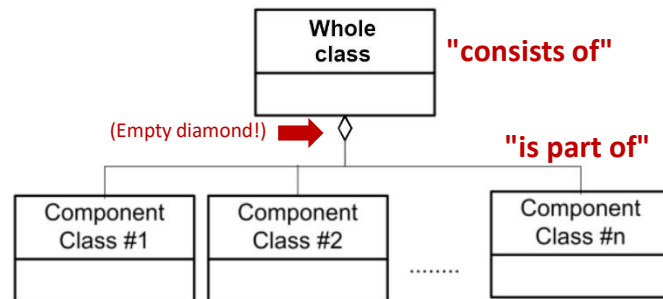
Aggregation represents a *weak* form of "has-a" or "whole/part" relationship between the aggregate (the whole) and its parts.

- The life cycle of the parts is not tightly coupled to the life cycle of the aggregate. If the aggregate is destroyed, its parts can continue to exist
- The contained objects can be part of multiple aggregates

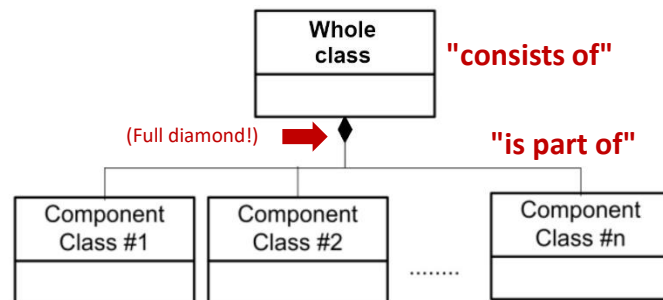
Composition represents a *strong* form of the "has-a" or "whole/part" relationship, characterized by a strong ownership and a coincidental lifecycle between the whole and its parts.

- The whole is responsible for the creation and destruction of its parts. The lifecycles of the parts are tightly coupled with the lifecycle of the whole.
- Each part is contained in only one whole at a time and cannot be shared among multiple wholes

Aggregation between classes

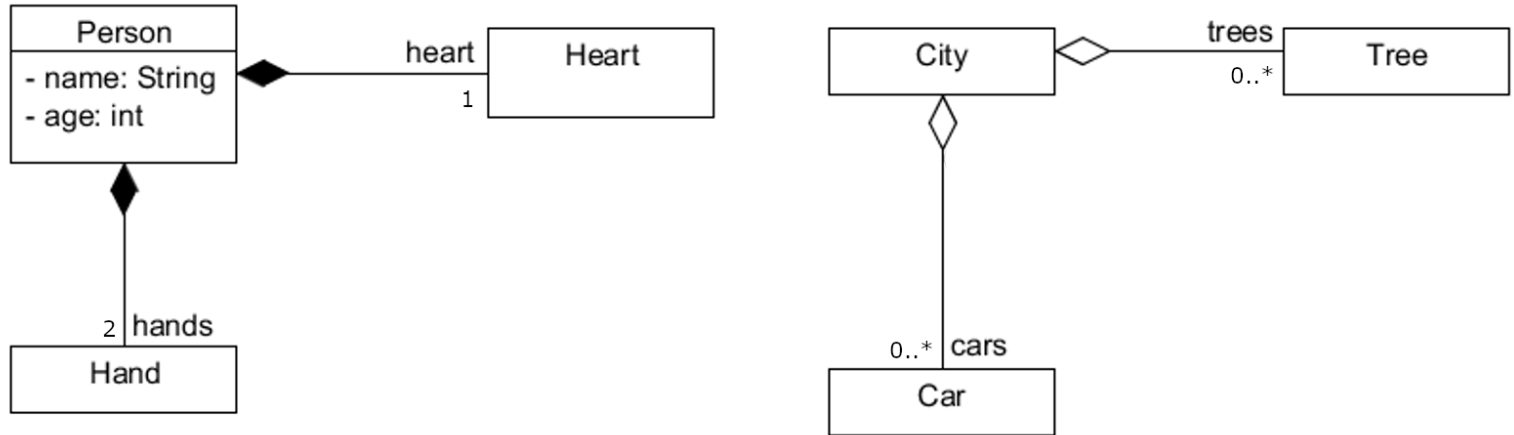


Composition between classes



UML in a nutshell

- Examples of **composition** (left) and **aggregation** (right) between classes





explained with...

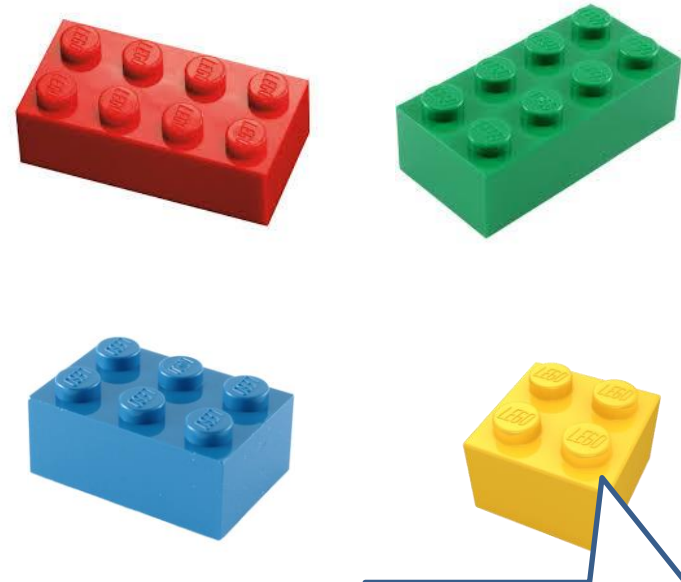


Class

«FeatureType» LegoBrick
<ul style="list-style-type: none"> + name: CharacterString [0..*] + fabricationYear: int + description: CharacterString [0..1] + inProduction: boolean + productionDate: Date + width_A_studs: int + width_B_studs: int + height: Length + width_A: Length + width_B: Length + geometry: GM_Solid + colour: BrickColourTypeValue

«enumeration» BrickColourTypeValue
<ul style="list-style-type: none"> yellow red blue green grey white black

Objects *(Instances of the class)*



FYI: This a Lego "stud"

Attributes / Properties

Attributes / Properties **type**:
Can have a simple or
complex structure.
E.g. length = double (for the
value) + CharacterString (for
the Unit of Measure)

Enumerations contain
closed sets of possible
values
(**Codelists** contain *open* sets
of possible values)

«FeatureType» LegoBrick

```
+ name: CharacterString [0..*]  
+ fabricationYear: int  
+ description: CharacterString [0..1]  
+ inProduction: boolean  
+ productionDate: Date  
+ width_A_studs: int  
+ width_B_studs: int  
+ height: Length  
+ width_A: Length  
+ width_B: Length  
+ geometry: GM_Solid  
+ colour: BrickColourTypeValue
```

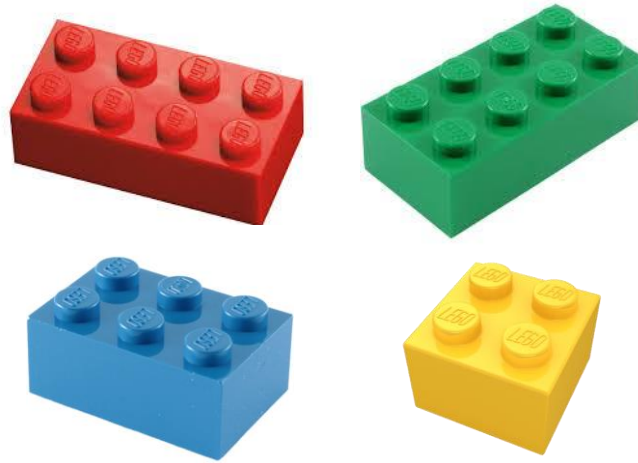
Multiplicity

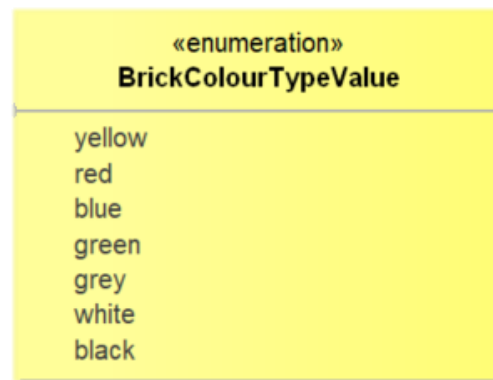
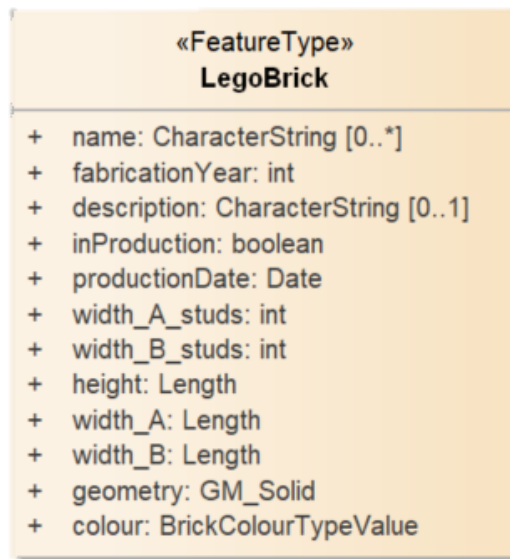
«enumeration» BrickColourTypeValue

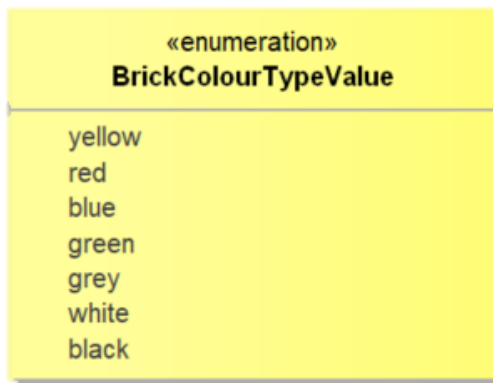
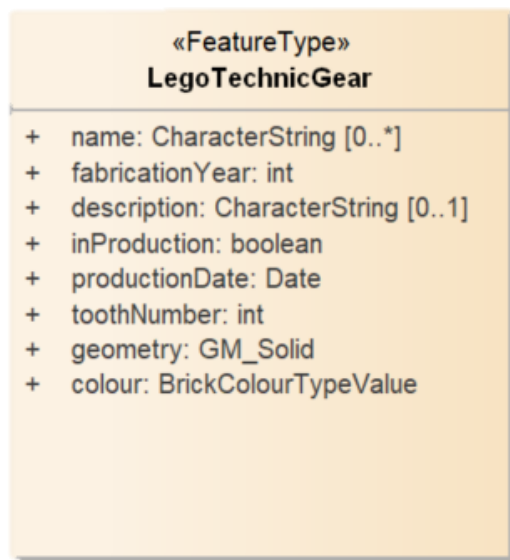
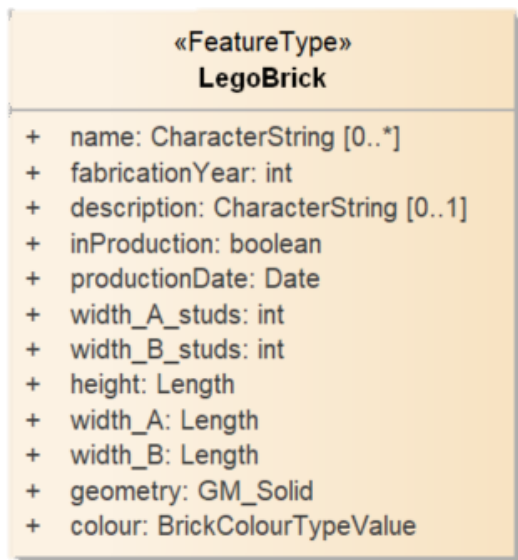
yellow
red
blue
green
grey
white
black

How to represent in UML that Lego does not produce only bricks, but also other elements?

Intro
Classes
 Relations
 Packages
 Namespaces

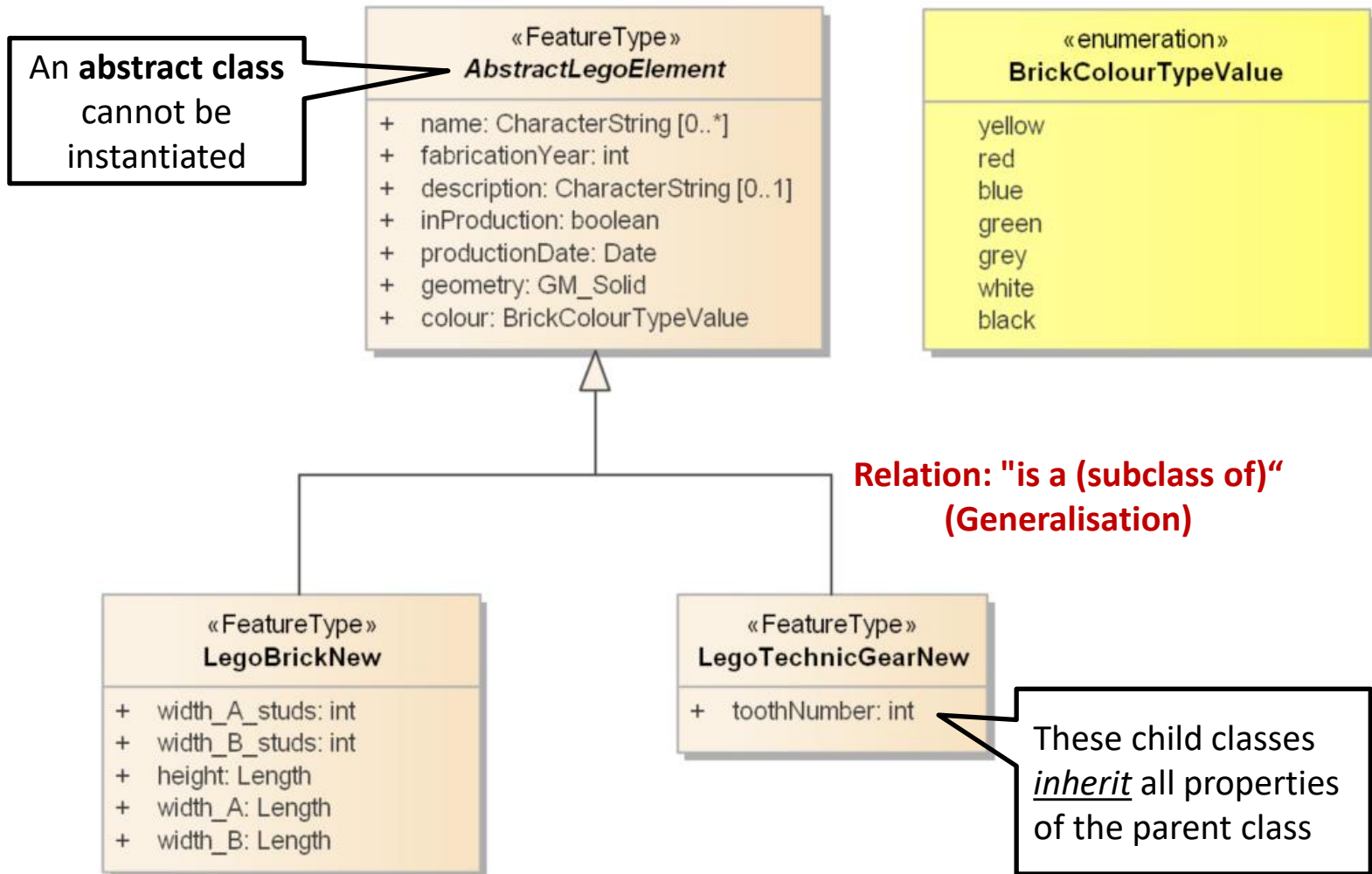


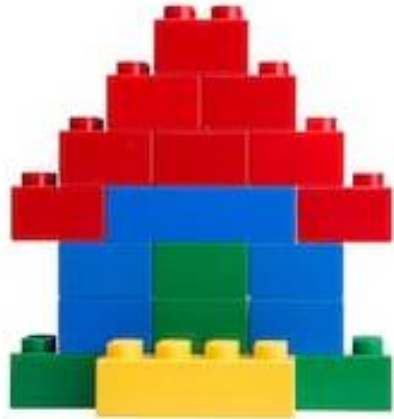




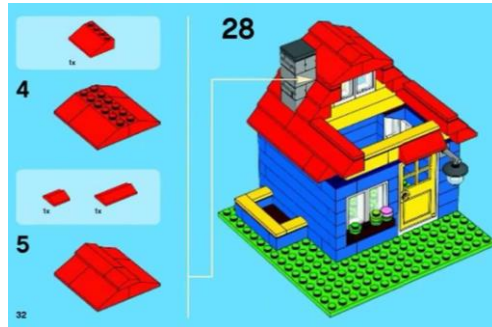
We can simply add another class. However, the two classes share some properties.

Is there a more elegant way to model them?



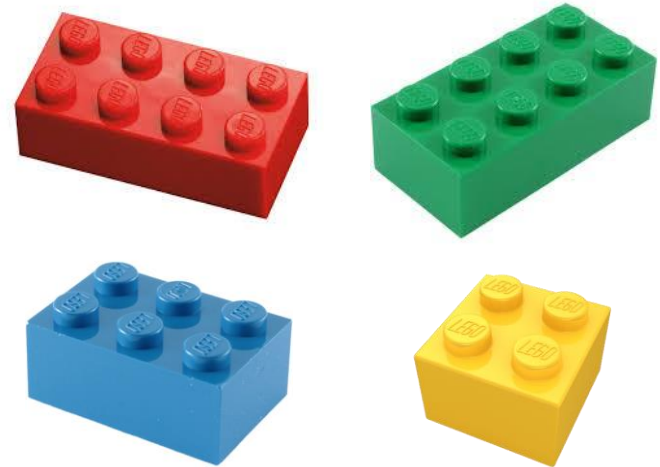


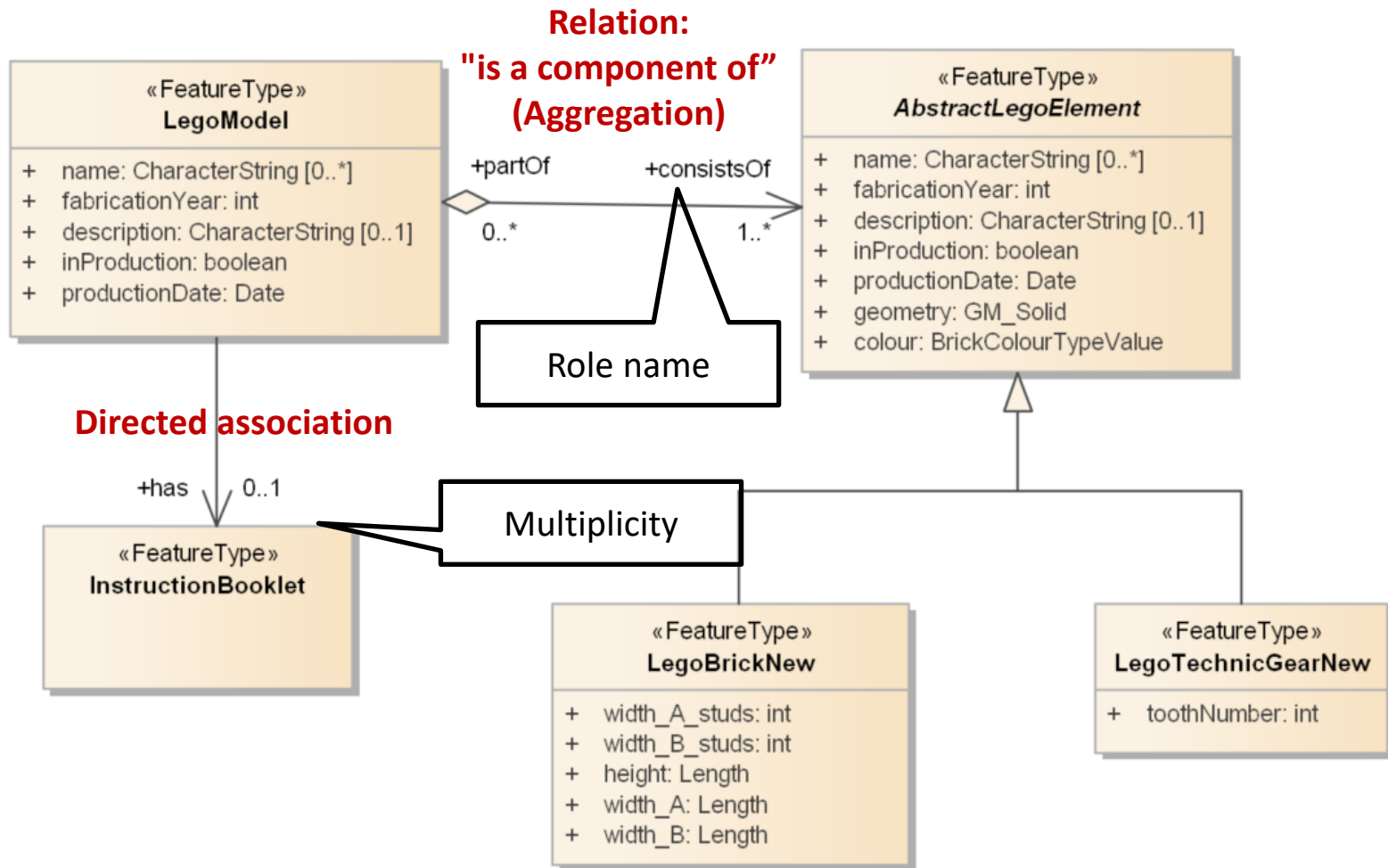
shutterstock.com • 300319532



How to represent in UML that:

- 1) a Lego model (e.g. a house) is made of Lego bricks, and
- 2) a set of building instructions is generally also included when you buy it?







CREATOR

Ages/edades

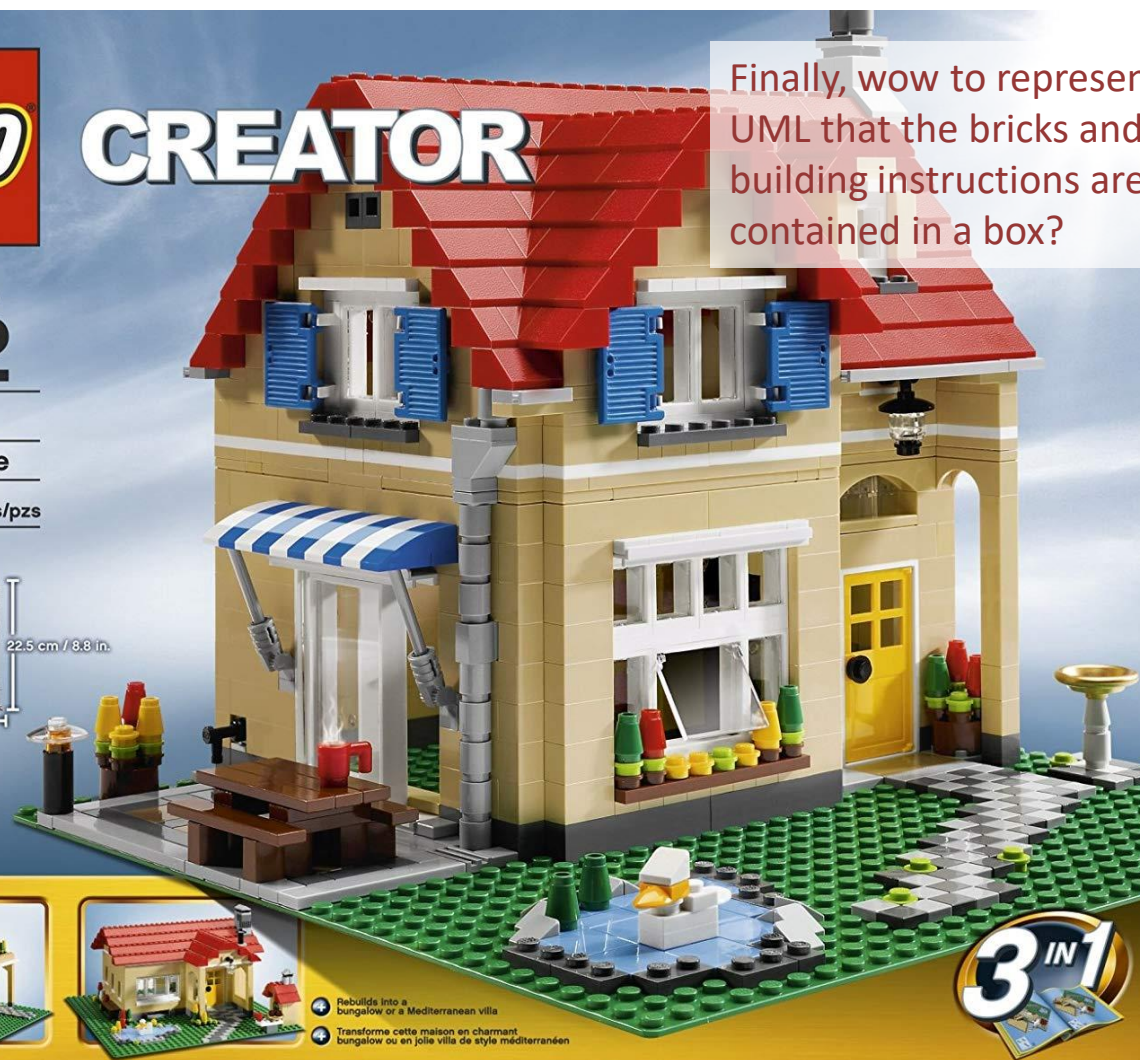
8-12

6754

Family Home

Cont. **976** pcs/pzs

Building Toy
Jouet de Construction
Juguete para Construir



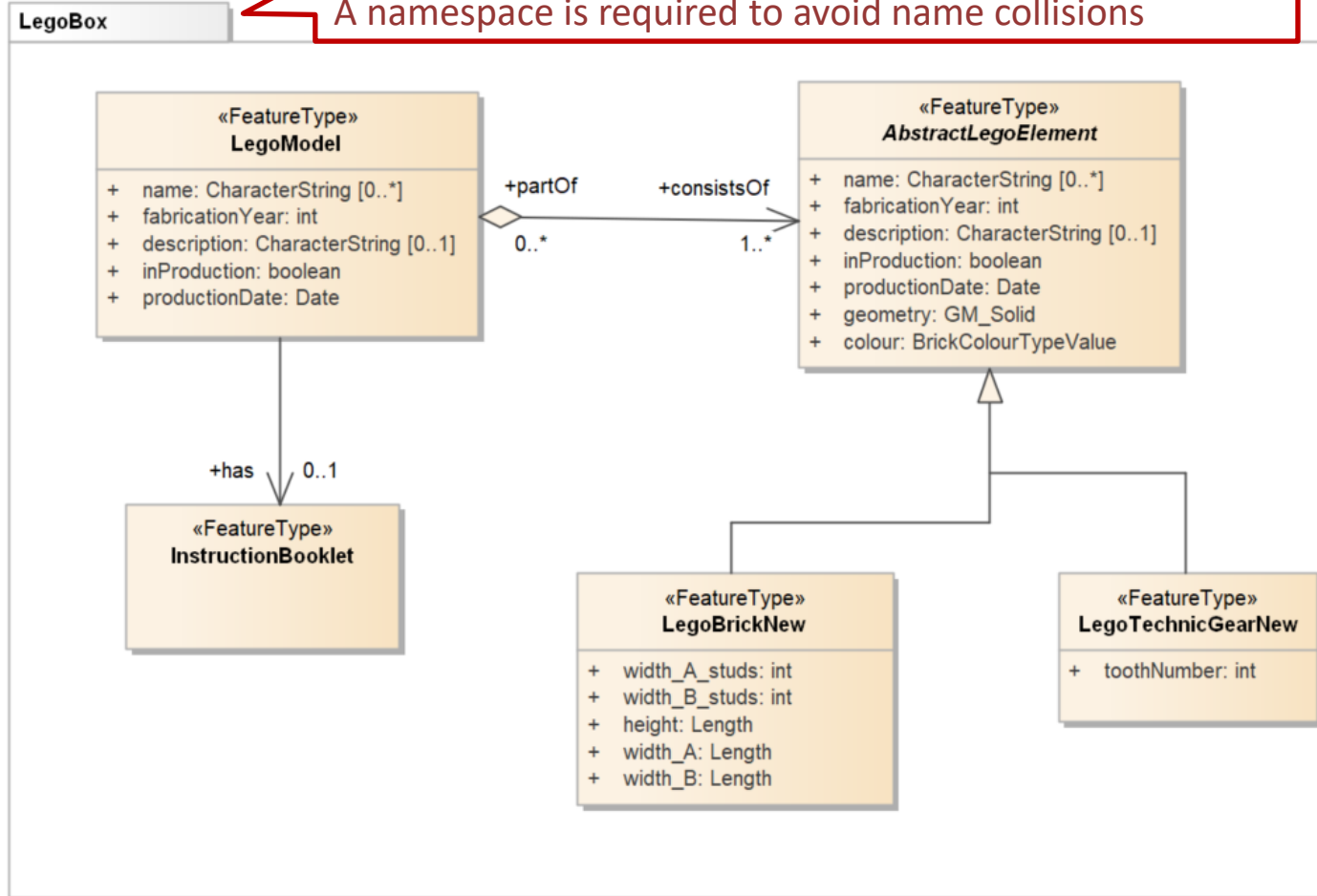
Finally, wow to represent in UML that the bricks and the building instructions are contained in a box?



- Rebuilds into a bungalow or a Mediterranean villa
- Transforme cette maison en charmant bungalow ou en jolie villa de style méditerranéen



A package groups elements that belong together.
It also provides a *namespace* for the grouped elements.
A namespace is required to avoid name collisions



Namespaces

- Imagine that we are modelling class "**EARTH**".
- Look at these are 3 examples of different meanings of the word "earth": if we create 3 classes, all called "Earth", we (or the computer) may not know anymore what is what...



Planet



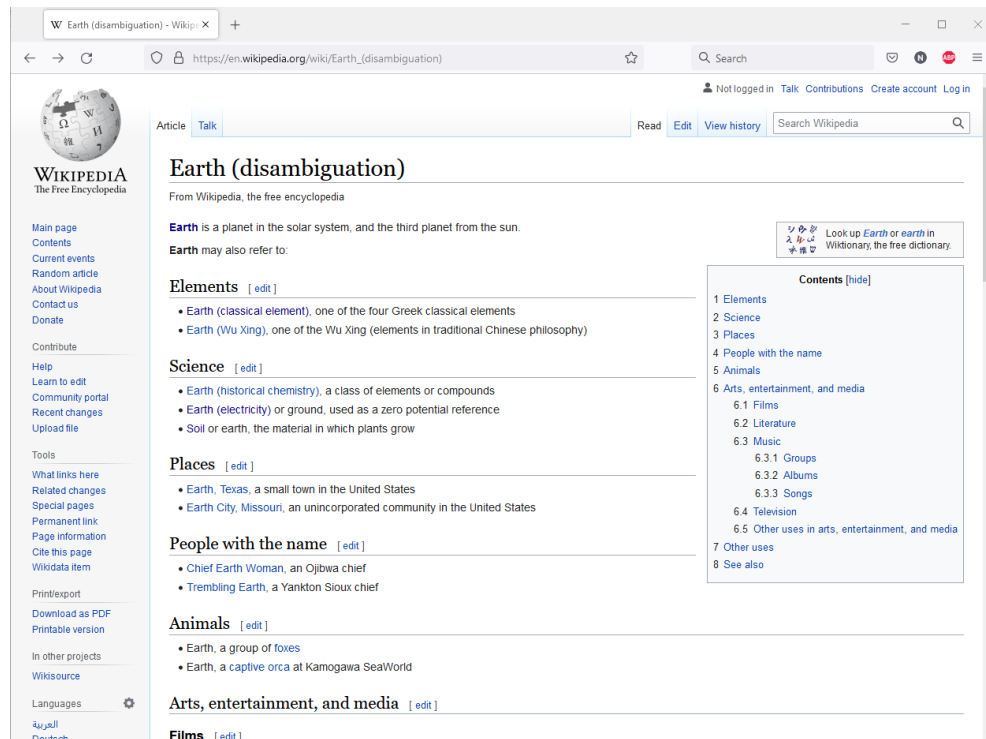
Soil



Electrical engineering

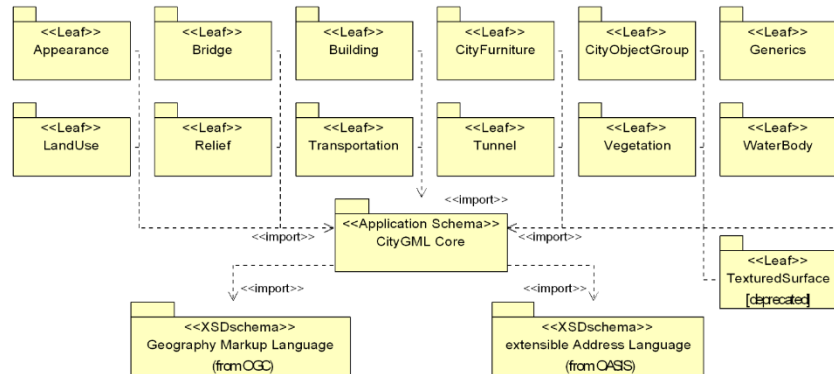
Namespaces

- Hence, we need a *namespace*!!
- You can think of a namespace as a sort of "surname" for a person. It helps reducing the cases of confusion with two or more people with the same name...
- So, we may end up with:
 - Astronomy:Earth
 - Geology:Earth
 - Electricity:Earth



Namespaces

- In UML, the concept of package is used to represent the namespace
- All classes within the same package are intended to belong together and share some common characteristics
- In the case of class earth example, you'd need three different packages in order to separate the 3 different earth classes and avoid semantic misunderstanding
- In CityGML, for example, there are several packages. The one called "Building" contains all classes to model a building, the one called "Terrain" contains all classes to model a digital representation of the terrain, etc.



Namespaces

- In an XML document, the definition of the namespaces is given in the header, using the tag **xmlns** (stands for... xml namespace ☺)
- For classes (and their properties/attributes) it is then indicated before the name using a ":" (colon) as separator

```

iearch View Encoding Language Settings Tools Macro Run Plugins Window ?
[Icons]
output.gml
<?xml version="1.0" encoding="UTF-8"?>
<core:cityModel xmlns:bridge="http://www.opengis.net/citygml/bridge/2.0" xmlns:tran="http://www.opengis.net/citygml/transportation/2.0" xmlns:frn=
"http://www.opengis.net/citygml/cityfurniture/2.0" xmlns:wtr="http://www.opengis.net/citygml/waterbody/2.0" xmlns:sch=
"http://www.ascc.net/xml/schematron" xmlns:veg="http://www.opengis.net/citygml/vegetation/2.0" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:tun=
"http://www.opengis.net/citygml/tunnel/2.0" xmlns:tex="http://www.opengis.net/citygml/texturedsurface/2.0" xmlns:gml="http://www.opengis.net/gml"
xmlns:gen="http://www.opengis.net/citygml/generics/2.0" xmlns:dem="http://www.opengis.net/citygml/relief/2.0" xmlns:app=
"http://www.opengis.net/citygml/appearance/2.0" xmlns:luse="http://www.opengis.net/citygml/landuse/2.0" xmlns:xAL=
"urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:smil20lang=
"http://www.w3.org/2001/SMIL20/Language" xmlns:pbase="http://www.opengis.net/citygml/profiles/base/2.0" xmlns:smil20="http://www.w3.org/2001/SMIL20/"
xmlns:bldg="http://www.opengis.net/citygml/building/2.0" xmlns:core="http://www.opengis.net/citygml/2.0" xmlns:grp=
"http://www.opengis.net/citygml/cityobjectgroup/2.0">
  <gml:description>fdfsdfdsfdfsdfsd</gml:description>
  <gml:name>test_citymodel</gml:name>
  <gml:boundedBy>
    <gml:Envelope srsName="urn:ogc:def:crs,crs:EPSG::31256,crs:EPSG::5176" srsDimension="3">
      <gml:lowerCorner>-1092.9130125475285 336988.13746829785 3.021538019180298</gml:lowerCorner>
      <gml:upperCorner>-438.3911874524715 337572.09303170216 90.48</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <app:appearanceMember>
    <app:Appearance gml:id="App 1">
  </app:appearanceMember>
  <app:appearanceMember>
  </app:appearanceMember>
  <core:cityObjectMember>
    <dem:ReliefFeature gml:id="ReliefFeature UUID bfdf1be8-ce74-43bd-b025-205e539bfa79">
  </core:cityObjectMember>
  <core:cityObjectMember>
    <tran:Road gml:id="Road UUID 2dedadef-ab8a-49bc-b03f-aab0aac1363a">
      <gml:description>This is a single object representing the whole street network</gml:description>
      <tran:lodiMultiSurface>
  </tran:Road>
  </core:cityObjectMember>
  <core:cityObjectMember>

```

Thank you for your attention!



Dr. Giorgio Agugiaro

g.agugiaro@tudelft.nl

3D Geoinformation Group

TU Delft

The Netherlands

<https://3d.bk.tudelft.nl/gagugiaro>