UML Class Diagram

UML Class Diagram Example

```
<<persistent>>
Project

+ pid: char
+ location: string = "Phoenix"
+ cost: real

<<create>>
+ newProject();

<<destroy>>
+ removeProject();

<<standardAccess>>
+ getPID();
# setPID();
+ getLocation();
# setLocation(string location);
+ getCost();
# setCost(real cost);
```
Details

Visibility
+publicAttribute, -privateAttribute, #protectedAttribute

Stereotype
<<persistent>>, <<create>>, ...

Attribute syntax
[visibility] name [multiplicity] [: type] [= initial-value] [{property-string}]

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

where each parameter has the form
[direction] name : type [= default-value]

and directions can be - in, out, inout.

Connecting classes

There are 3 types of connections in UML (Inter-class relationships)

• **Generalizations** which are *inheritance* relationships.

• **Associations** which are structural relationships: Represents relationships between instances of two classes.
  • Aggregation and Composition
  • **Dependencies** which are *using* relationships.
Relationships: 3 Kinds

- **Generalization**
  - Relationship between general thing (parent) and more specific thing (child)
  - Child “is-a-kind-of” parent.
  - Child inherits attributes and operations of parent.
  - Use when you want to show parent child relationship
Associations (UML)

- Represent conceptual relationships between classes (structural)
  - Customers make Orders
  - Professors teach Courses

Sometimes directed arrows are used to specify navigability.

Associations - In Other OOAD

Associations may be binary, ternary, or higher order

*binary association*

*Fusion Style*

*Ternary association*
**Associations – UML Links**

– Link is a semantic connection among objects.
– A link is an instance of an association.

**Fig. 1: UML diagram for associations.**

**Associations – Link Attributes**

- **Link Attributes**

The most compelling reason for having links and attributes is for many-to-many relationships.

**Fig. 2: UML diagram for link attributes.**
**Associations – Link Attributes**

- **UML Association Class**
- An association class is used when a relationship between objects has *properties*.

![UML Association Class Diagram](image)

**Associations - Aggregation**

- Structural association representing “whole/part” relationship.
- “has-a” relationship.

![UML Aggregation Diagram](image)
Aggregation and Composition

Composition is a stronger form of aggregation. Composite parts live and die with the whole.

Dependency

- A change in one thing may affect another.
- "Uses" relationship.
- May have a name, but not common.
- Use Dependency to show one thing uses another.
**Modeling Structural Relationships**

- Considering a bunch of classes and their association relationships

![UML Diagram](uml_diagram.png)

**Dependencies**

A dependency shows that one class **uses** another. A change in one will affect the other.

```
Account
```

```
Money
```

- method parameters
Abstract classes, Types, Interfaces

- **«type»** SomeType
  - Types have no implementation
  - Used for built in types like int.

- **«interface»** SomeFace
  - Interfaces only have operations
  - Java’s interfaces have attributes too.

- **{abstract}** SomeClass
  - Abstract classes have no instances
  - Strictly speaking, these map to Java interfaces.

Interfaces and Realization

- Frame
- ActionListener
  - `action performed()`
Abstract classes

![UML class diagram for abstract classes]

Generics (Parameterized Types)

![UML class diagram for generics]

UML class diagrams 19

UML class diagrams 20
**Association qualifiers**

- A *qualifier* is an association attribute that is unique within a set of related objects.
- A qualifier is used within a qualified association to relate a qualified object to a target object using a qualifier value from a set of qualifier values.

![Diagram showing association between Show and Ticket with qualifier](image)

**Recursive associations**

![Diagram showing recursive association between Queue Element](image)
Packages

- A package is a general-purpose mechanism for organising elements into groups.
- Packages may contain other packages.
- Dependencies between packages can be shown.
- There are visibility rules for package components.

Object instances

- `attendee: Person` (Named Instance)
  - `: Frame` (Anonymous Instance)
- `: Card` (MultiObject)
  - `steve: Person` (Object attributes)
    - `shoeSize = 42`
### Summary:
**Structural Modeling: Core Relationships**

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>association</td>
<td>a relationship between two or more classifiers that involves connections among their instances.</td>
<td></td>
</tr>
<tr>
<td>aggregation</td>
<td>A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.</td>
<td></td>
</tr>
<tr>
<td>generalization</td>
<td>a taxonomic relationship between a more general and a more specific element.</td>
<td></td>
</tr>
<tr>
<td>dependency</td>
<td>a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).</td>
<td></td>
</tr>
</tbody>
</table>

Reference: OMG tutorial on UML by Cris Kobryn

### Structural Modeling: Core Relationships (cont’d)

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>realization</td>
<td>a relationship between a specification and its implementation.</td>
<td></td>
</tr>
</tbody>
</table>

Reference: OMG tutorial on UML by Cris Kobryn