COMP 3111
SOFTWARE ENGINEERING

LECTURE 4
MODELING SOFTWARE SYSTEMS
USING UML
EXERCISE

EXERCISE: GRAPHICAL EDITOR

We want to represent information about a graphical editor that supports grouping. A document is composed of several sheets. Each sheet can contain several drawing objects, which can be text, geometric objects or groups. A group is simply a collection of drawing objects, possibly including other groups. A group contains two or more drawing objects. A drawing object can be a member of at most one group. Each sheet belongs to at most one document and each drawing object belongs to exactly one sheet.

Construct a class diagram that shows, as necessary, the required classes, any associations, aggregations, compositions and generalizations among the classes as well as any necessary association classes and constraints. Associations should be named, if necessary. Using the problem statement and real-world knowledge, give the most likely multiplicity for each association, aggregation and composition. If a multiplicity cannot be inferred from the problem statement or real-world domain knowledge, then indicate this with a "?".

EXERCISE: GRAPHICAL EDITOR—SOLUTION

Analysis of Problem Statement

Since we only need to show how the classes are related, the analysis will focus only on discovering the associations, aggregations, compositions and generalizations among the classes and their related multiplicity.

We want to represent information about a graphical editor that supports grouping.

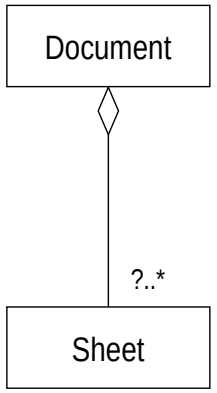
General statement about the problem domain.

A document is composed of several sheets.

aggregation: Sheet *PartOf* Document

multiplicity: max-card(Document, *PartOf*) = *

Remarks: A document can contain many (an unknown number) of sheets.



EXERCISE: GRAPHICAL EDITOR—SOLUTION

Each sheet can contain several drawing objects, which can be text, geometric objects or groups.

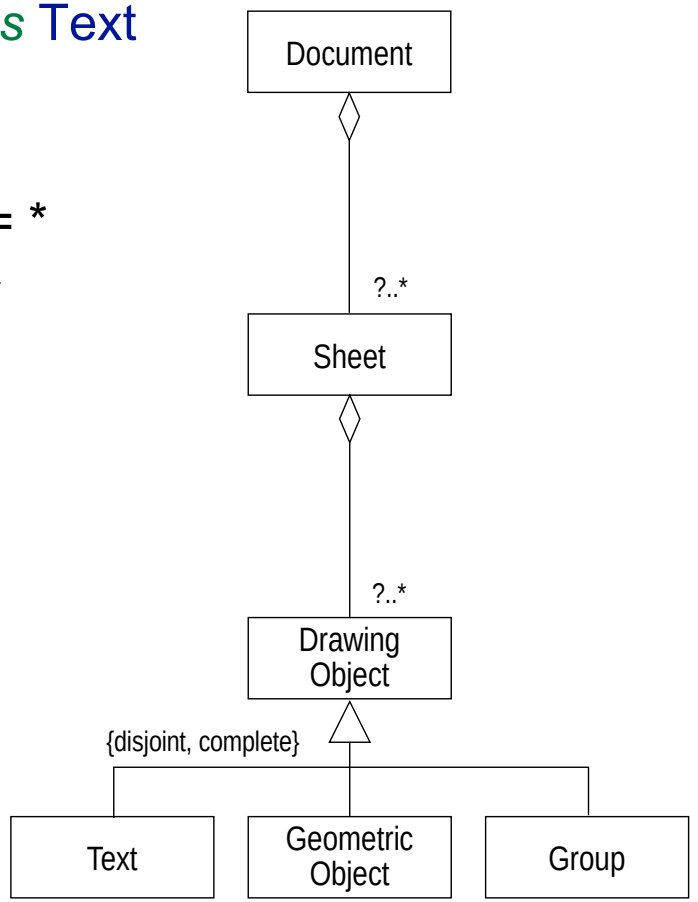
aggregation: DrawingObject *PartOf* Sheet

generalization: DrawingObject *generalizes* Text GeometricObject, Group

coverage: {disjoint, complete}

multiplicity: max-card(Sheet, *PartOf*) = *

Remarks: A sheet can contain many (an unknown number) of drawing objects.



EXERCISE: GRAPHICAL EDITOR—SOLUTION

A group is simply a collection of drawing objects, possibly including other groups.

aggregation: DrawingObject *PartOf* Group

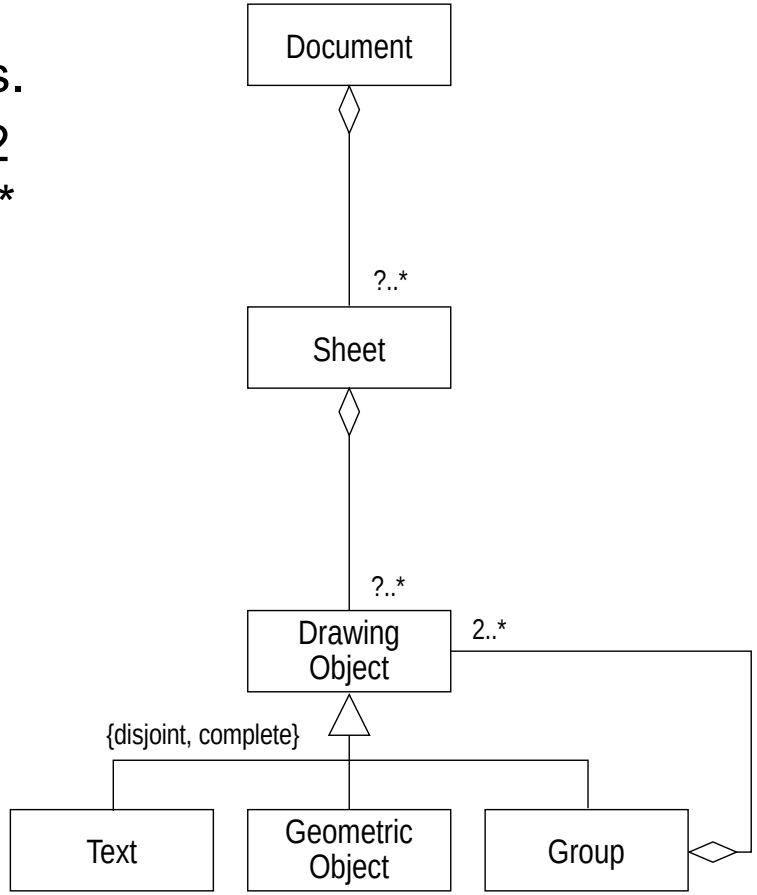
A group contains two or more drawing objects.

multiplicity: min-card(Group, *PartOf*) = 2
max-card(Group, *PartOf*) = *

A drawing object can be a member of at most one group.

multiplicity: max-card(DrawingObject, *PartOf(Group)*) = 1

Remarks: Already represented by aggregation association whose multiplicity is 0..1.



EXERCISE: GRAPHICAL EDITOR—SOLUTION

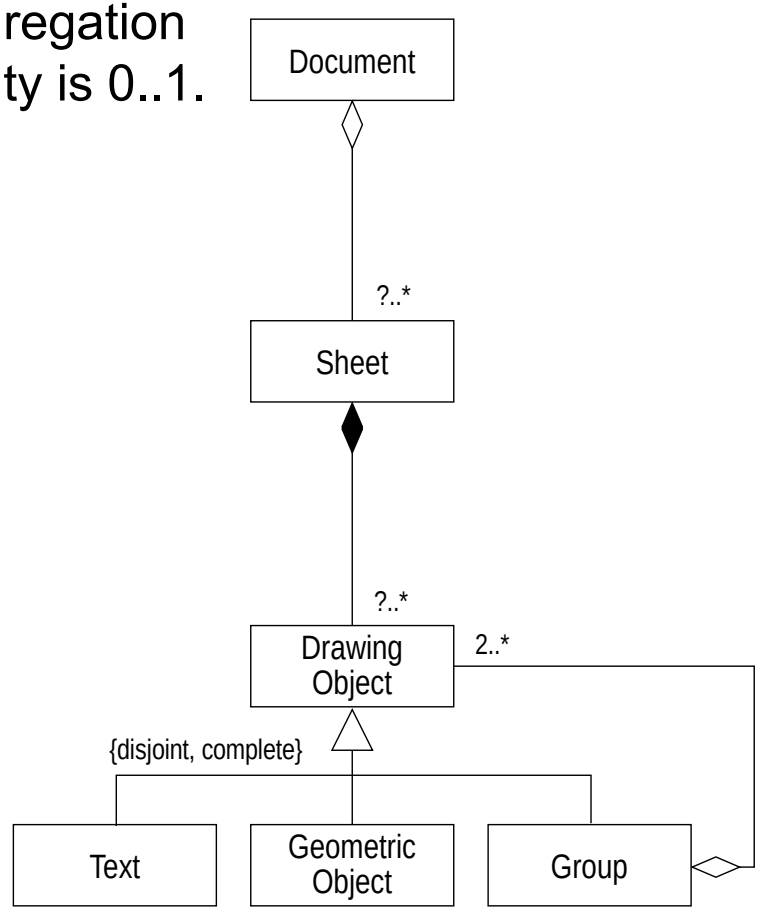
Each sheet belongs to at most one document
and each drawing object belongs to exactly one sheet.

multiplicity: max-card(Sheet, *PartOf(Document)*) = 1

Remarks: Already represented by aggregation association whose multiplicity is 0..1.

multiplicity: min-card(DrawingObject, *PartOf(Sheet)*) = 1
max-card(DrawingObject, *PartOf(Sheet)*) = 1

Remarks: Composition association whose multiplicity is 1..1.



EXERCISE: GRAPHICAL EDITOR—SOLUTION

Real World Knowledge

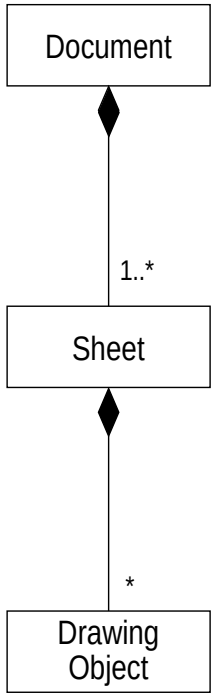
Multiplicity that can be reasonably inferred from real world knowledge.

A sheet must be contained in some document.

multiplicity: min-card(Sheet, *PartOf(Document)*) = 1

A document must contain at least one sheet;
else, there is no document.

multiplicity: min-card(Document, *PartOf*) = 1



A sheet can contain no drawing objects (i.e., it is empty).

multiplicity: min-card(Sheet, *PartOf*) = 0
Remarks: We could also assume that it must contain at least one drawing object.

A drawing object does not have to be a member of any group.

multiplicity: min-card(DrawingObject, *PartOf(Group)*) = 0
Remarks: Already represented by aggregation.

EXERCISE: GRAPHICAL EDITOR—SOLUTION

Class Diagram