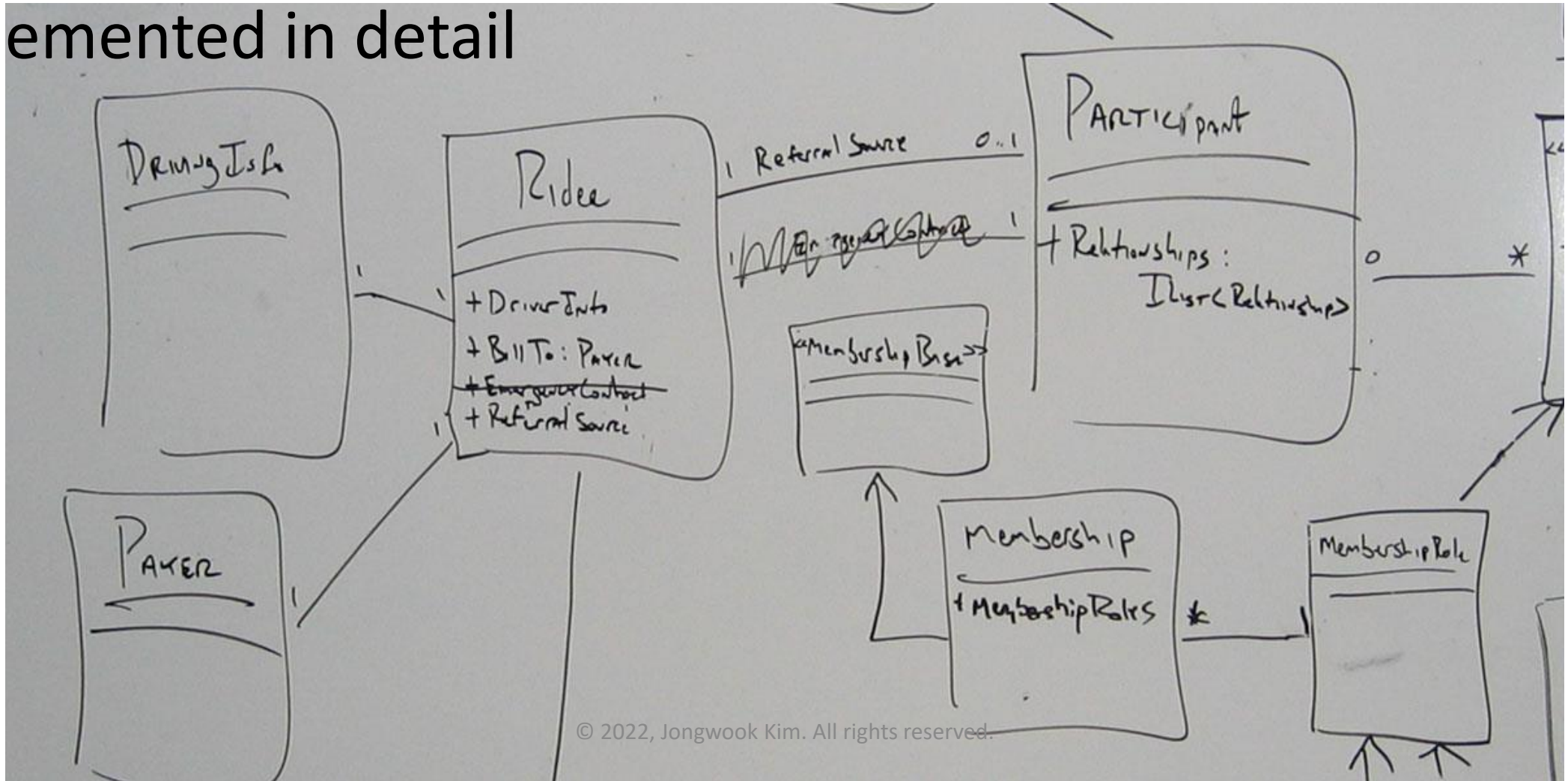
(3) Unified Modeling Language

Unified Modeling Language

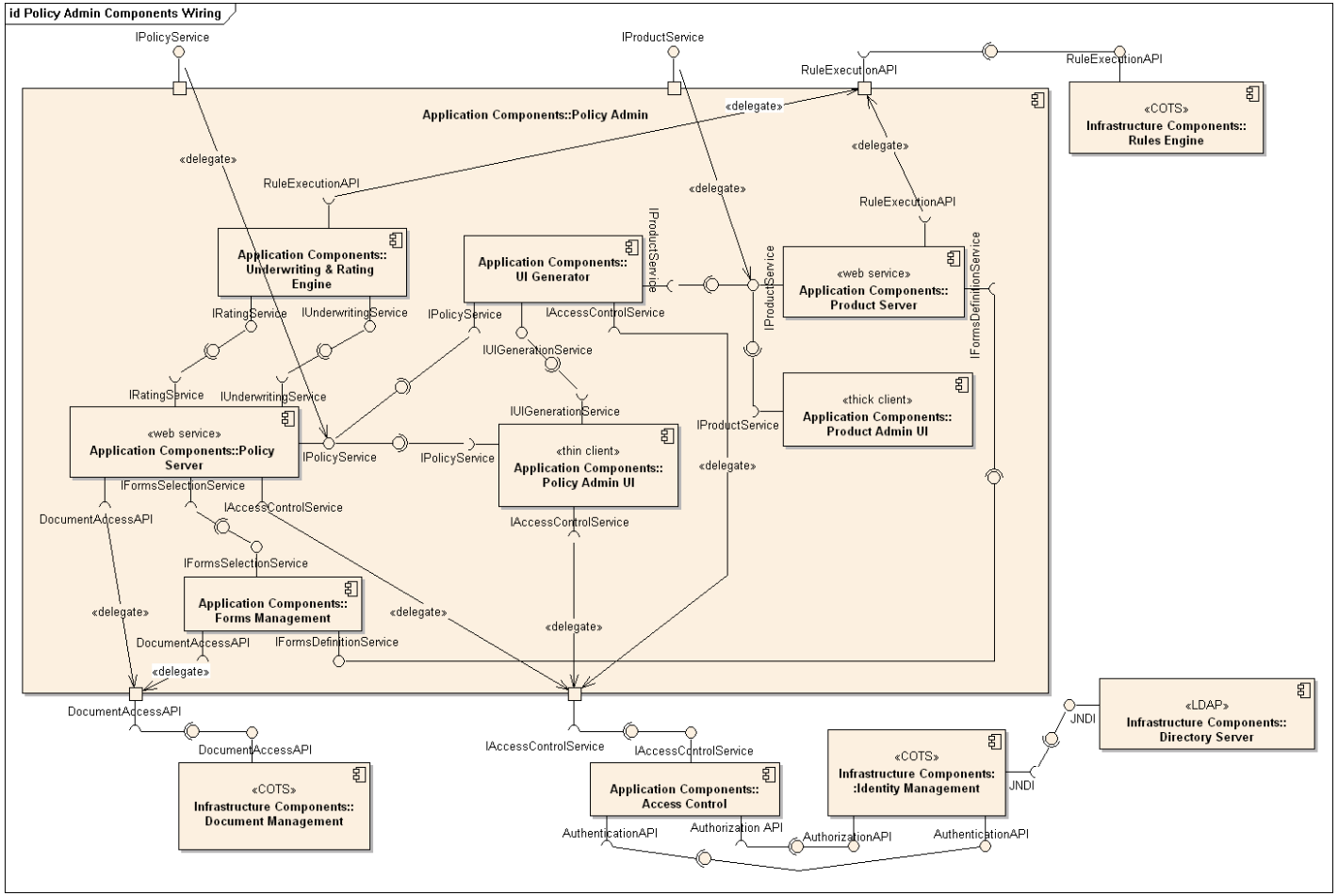
- A graphical modeling language for visualizing the structure and behavior of programs without specifying how they are implemented in detail



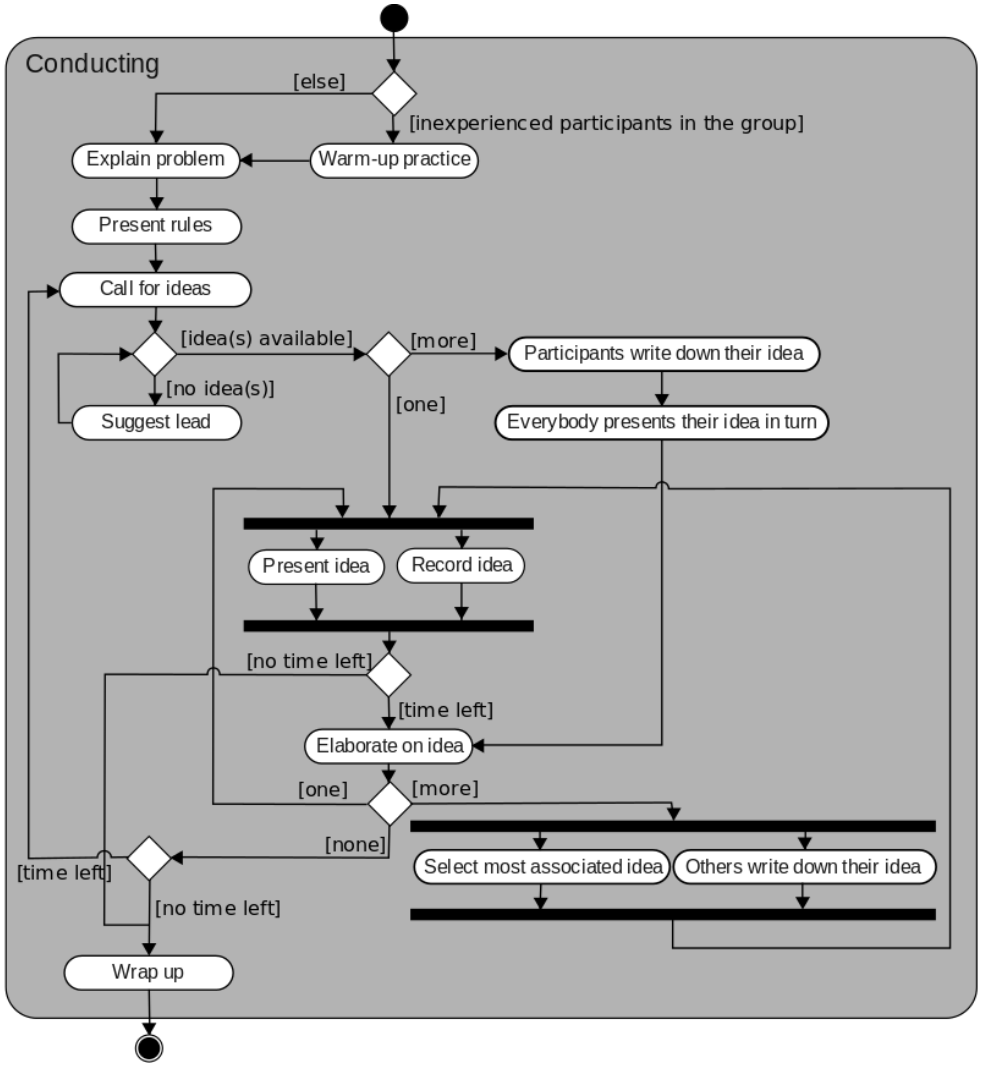
UML Diagram Types

- **Structural** UML Diagrams
  - Class diagram
  - Component diagram
  - Composite structure diagram
  - Deployment diagram
  - Object diagram
  - Package diagram
  - Profile diagram
- **Behavioral** UML Diagrams
  - Activity diagram
  - Communication diagram
  - Interaction overview diagram
  - Sequence diagram
  - State diagram
  - Timing diagram
  - Use case diagram

Examples



Component Diagram



Activity Diagram

UML Diagram Tools

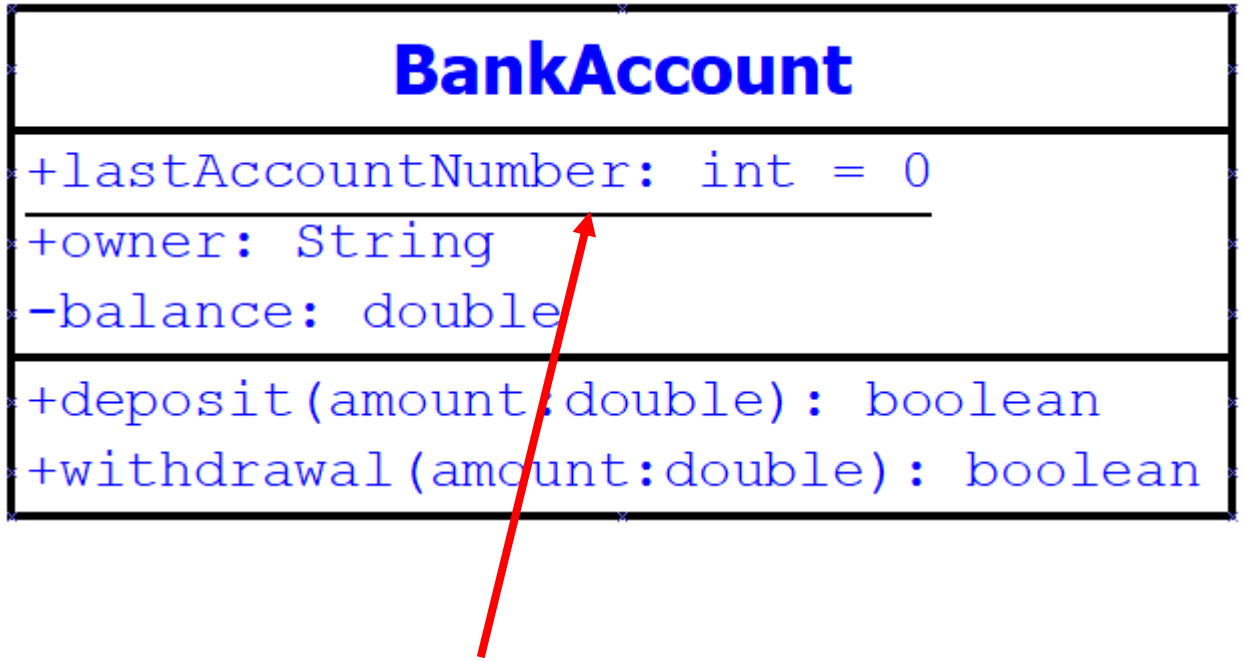
- We use “Dia Diagram Editor”

  http://dia-installer.de

Class Diagram

Class

- A class is drawn as three compartments:



← class name (required)

← attributes/fields (optional)

<visibility> <name> : <type>

← methods (optional)

<visibility> <name>(<param>*) : <return type>

<param> := <name> : <type>

static attributes (or methods) are underlined

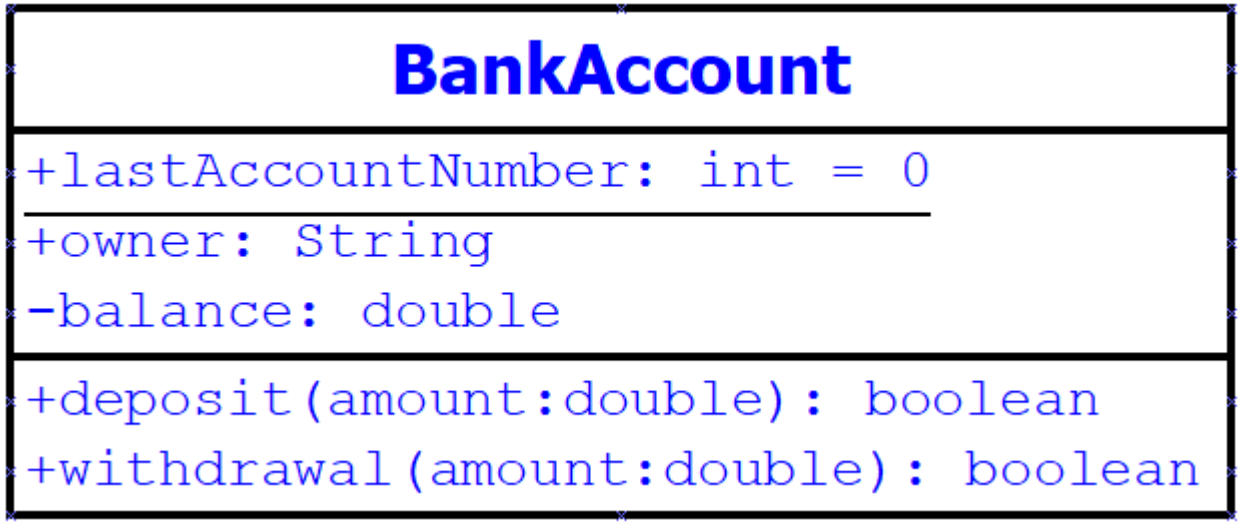
<visibility> can be one of:

– private

protected

+ public

Converting a Class Diagram into Java Code



⇒

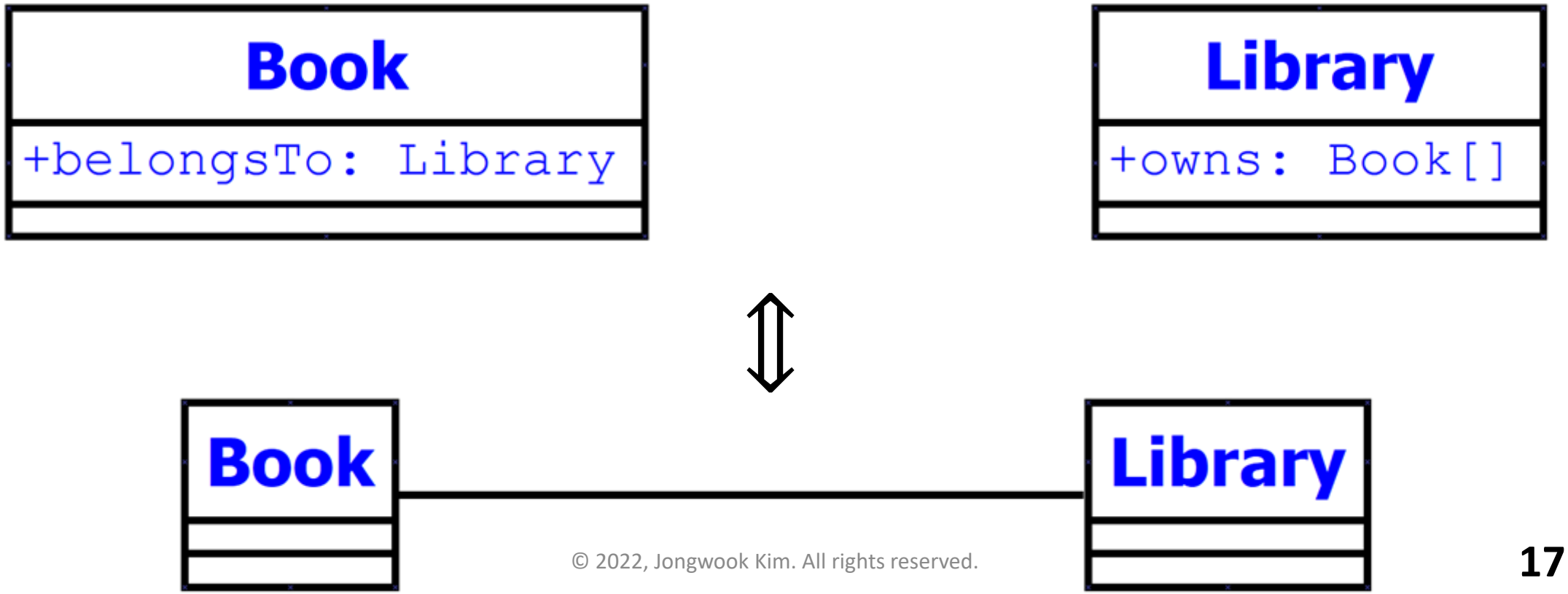
```
class BankAccount {

}
```

Instance Relationships

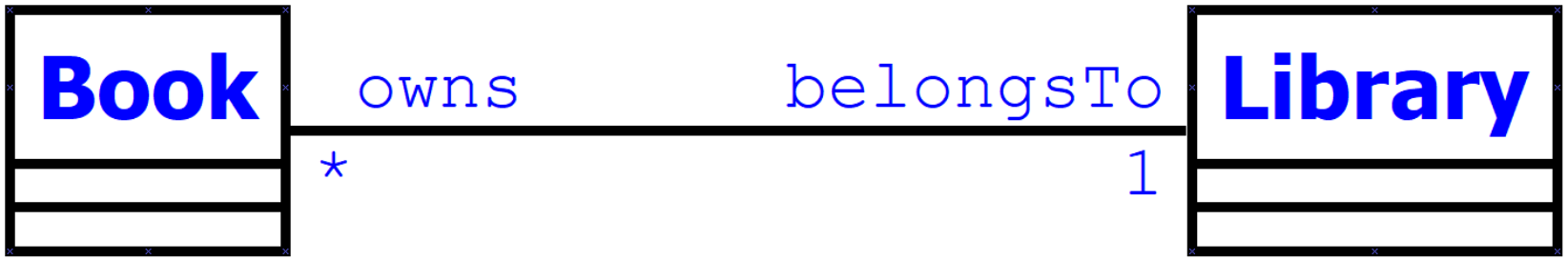
Association

- A has-a relationship between classes
  1. implicitly defines attributes of user-defined types
  2. is drawn as a line between classes



Association

3. has role names and allowable number of instances (called multiplicity)



- Each book belongs to one library
- Each library owns any number of books

4. An association can link any number of classes

Exercise

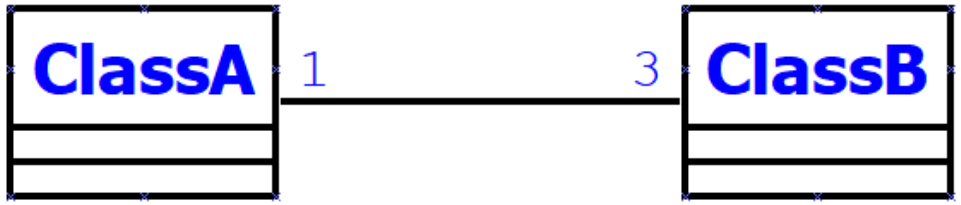
- Draw a class diagram that visualizes the following relationship
  - Each person owns any number of cars
  - Each car is owned by one person

1. Do not use association

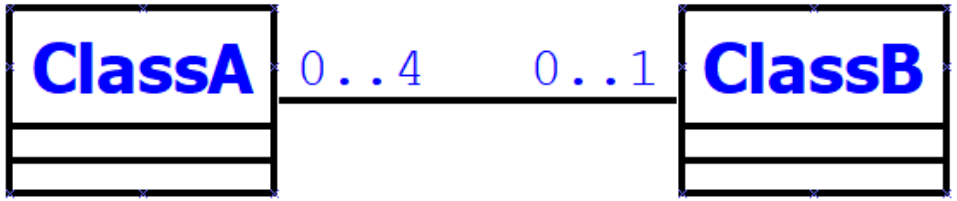
2. Use association

Multiplicity

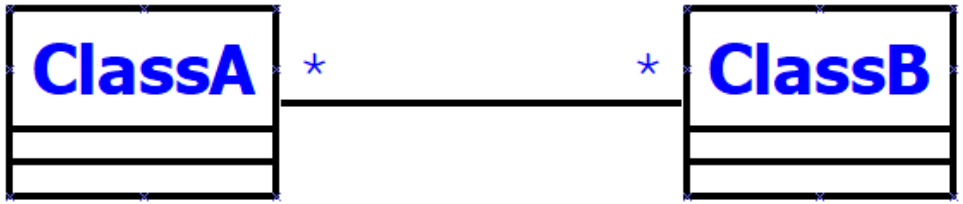
1. Exact number – just write it



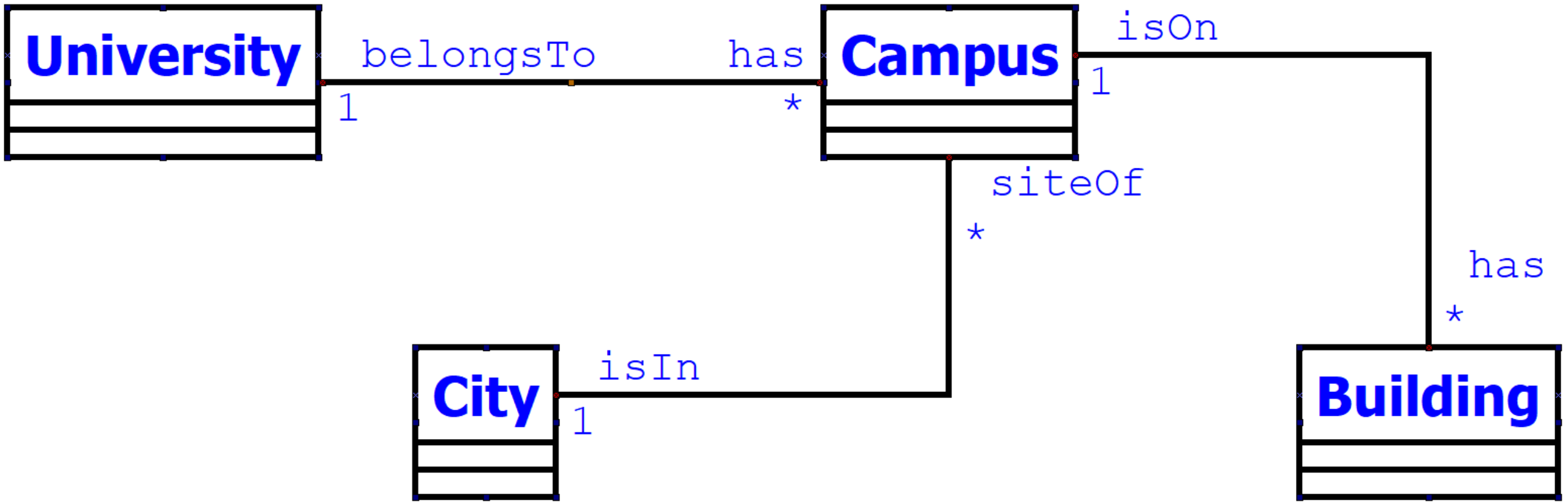
2. Range of numbers – two dots between numbers



3. Arbitrary number – write *



Example



Each campus belongs to 1 university
Each university has * campuses
Each campus is in 1 city
Each city is the site of * campuses
Each building is on 1 campus
Each campus has * buildings

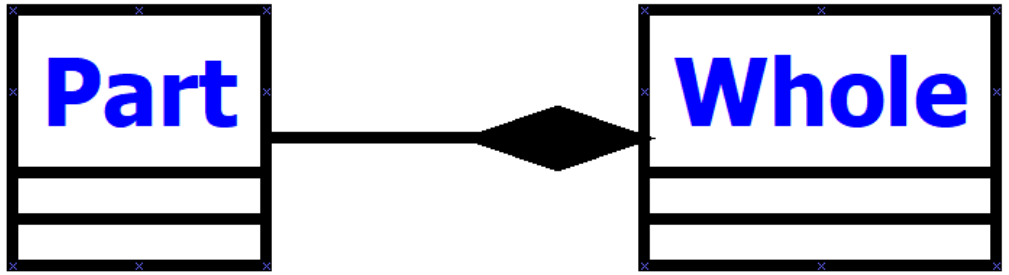
Each building is located in 1 city
Each city is the site of * buildings
Each university has * buildings
Each building belongs to 1 university
Each university is in * cities
Each city is the site of * universities

Exercise

- Each employee works for one company which can have 0 employees
- Each company has exactly one board-of-directors and vice versa
- Each office is allocated to zero or more employees, and an employee can have no office or at most one
- A person is the member of 0 or more board-of-directors (each board-of-directors has from 3 to 8 persons)

Composition

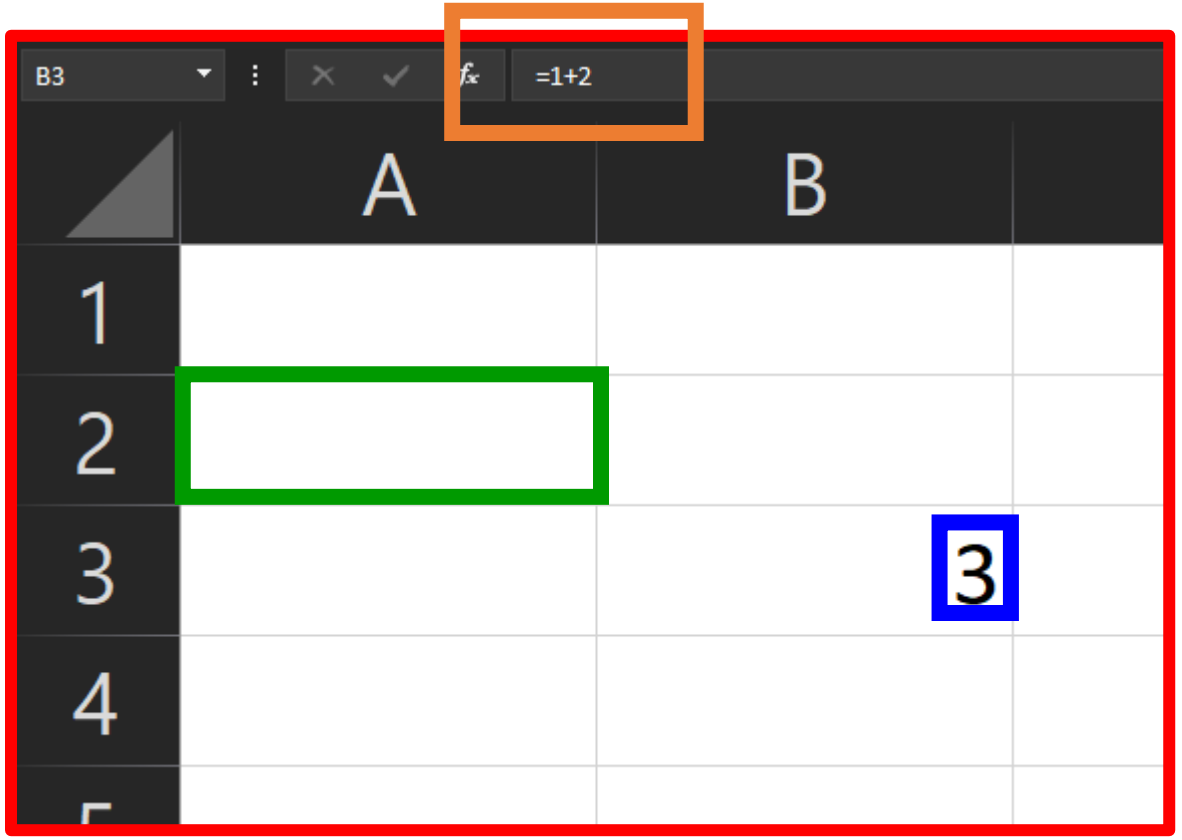
- Owns-relationship – an association variant
  - A **Whole**-type instance is the single owner of a **Part**-type instance



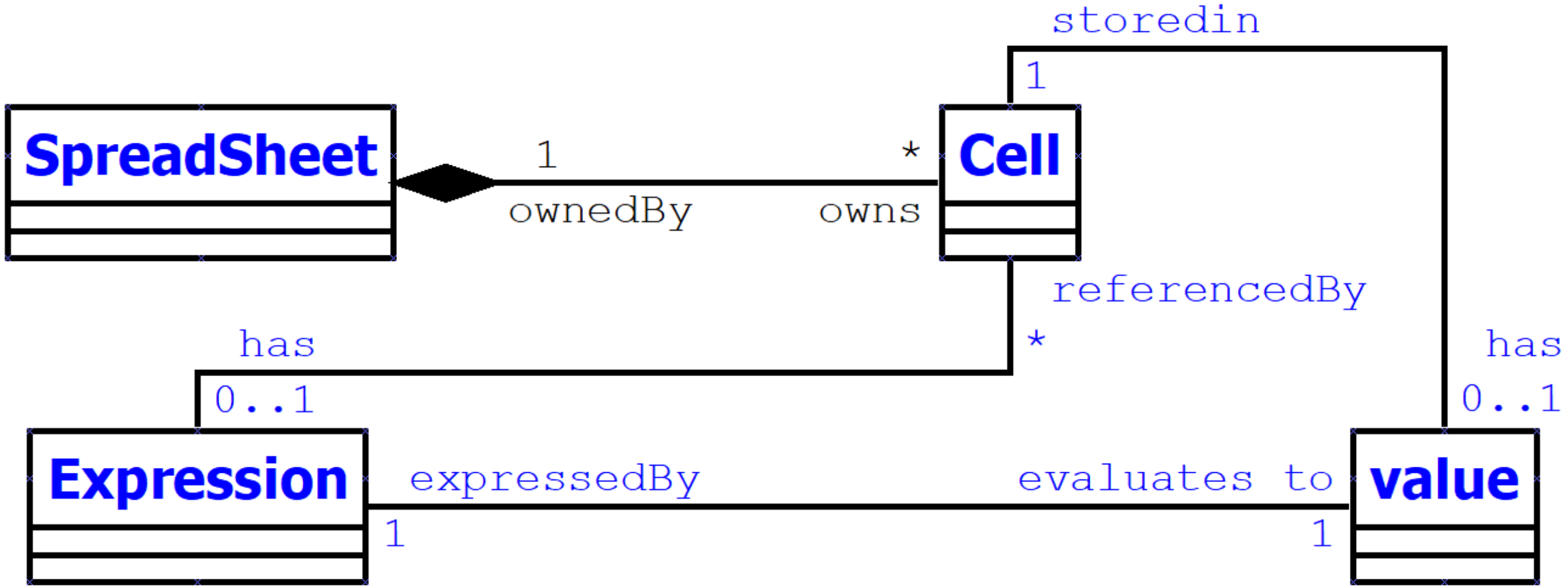

```
public class Whole {
    private Part p = new Part();
    ...
}
```

Exercise

- Draw a class diagram that visualizes the relationships between
  - SpreadSheet
  - Cell
  - Expression
  - Value

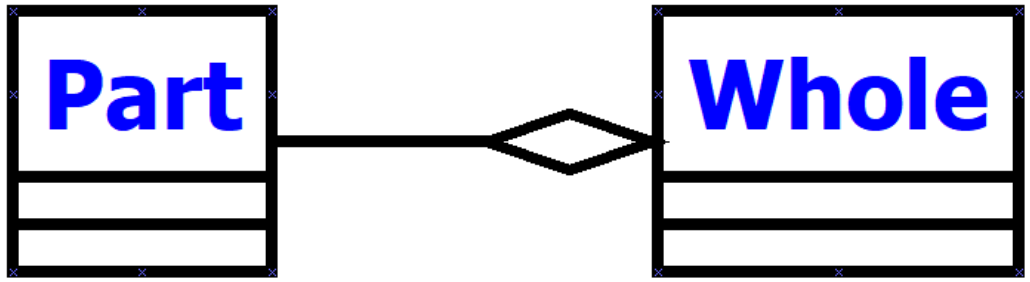


Answer



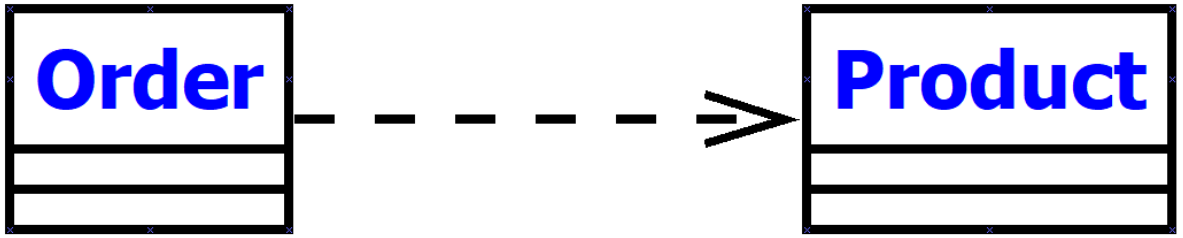
Aggregation

- Uses-relationship – another association variant
  - Existence of **Part**-type instances does not depend on the existence of **Whole**-type instances



Dependency

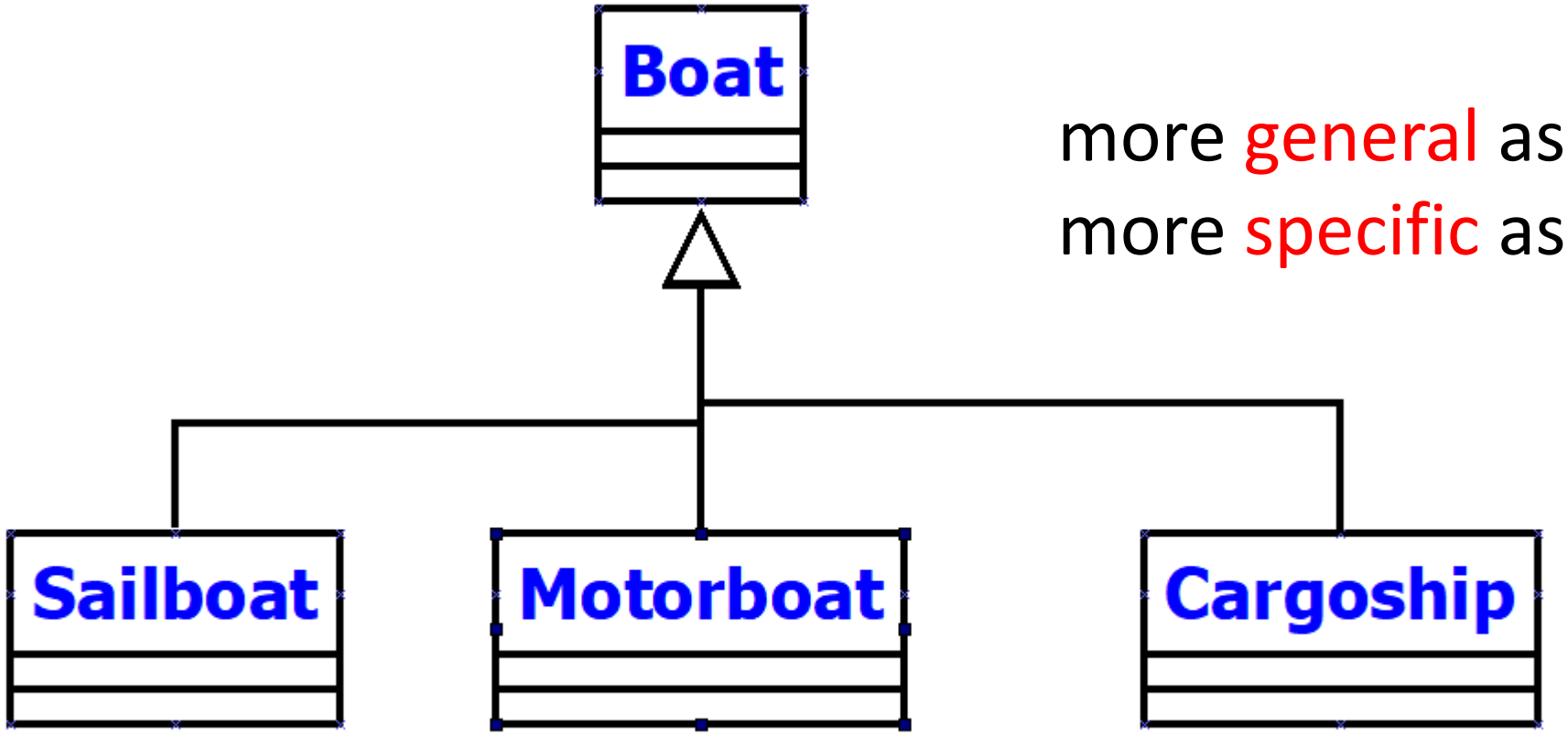
- Directed association relationship



Class Relationships

Generalization

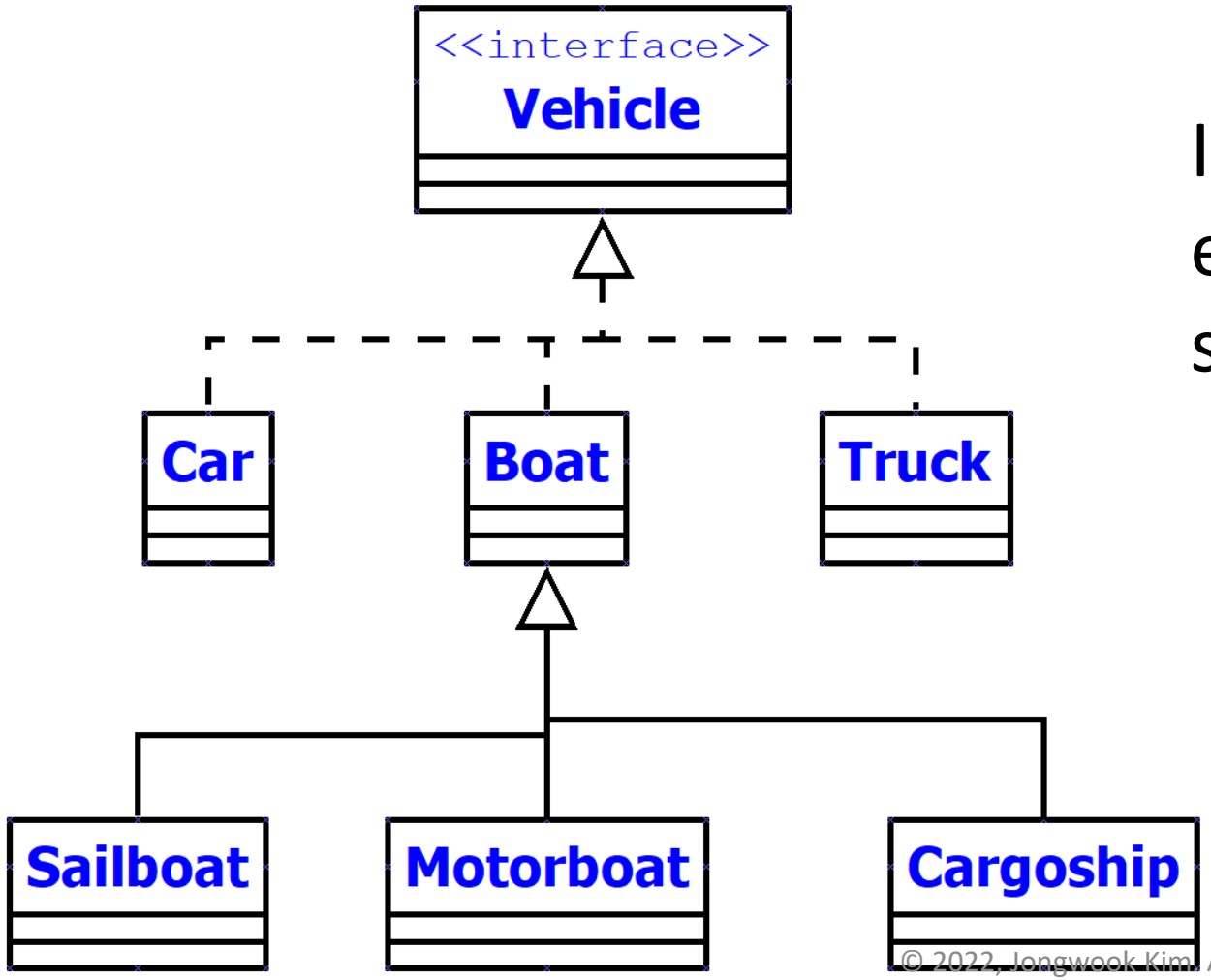
- An is-a relationship between classes (or interfaces)



more general as you move up
more specific as you move down

Realization

- An implementation relationship between a class and interfaces



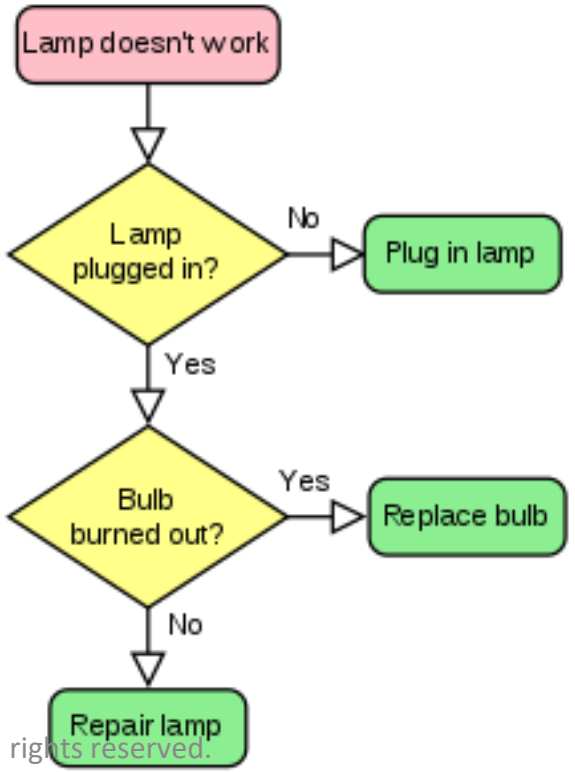
Interfaces drawn like classes except they are tagged by stereotypes **≪interface≫**

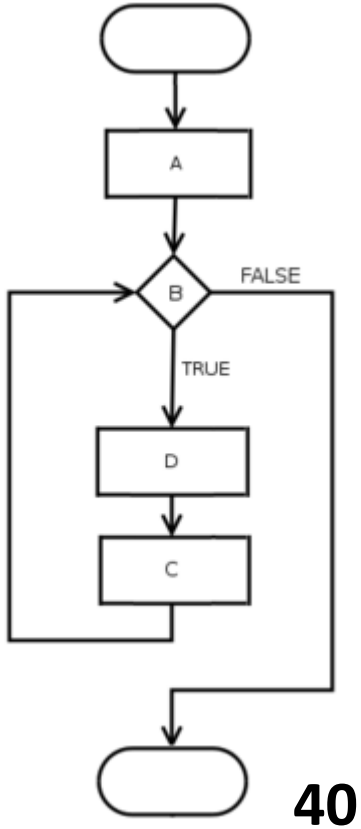
Exercise

- Draw a class diagram that visualizes the following relationship
  - Each classroom contains pieces of equipment
  - AV is a piece of equipment
  - Chair is a piece of equipment
  - Table is a piece of equipment

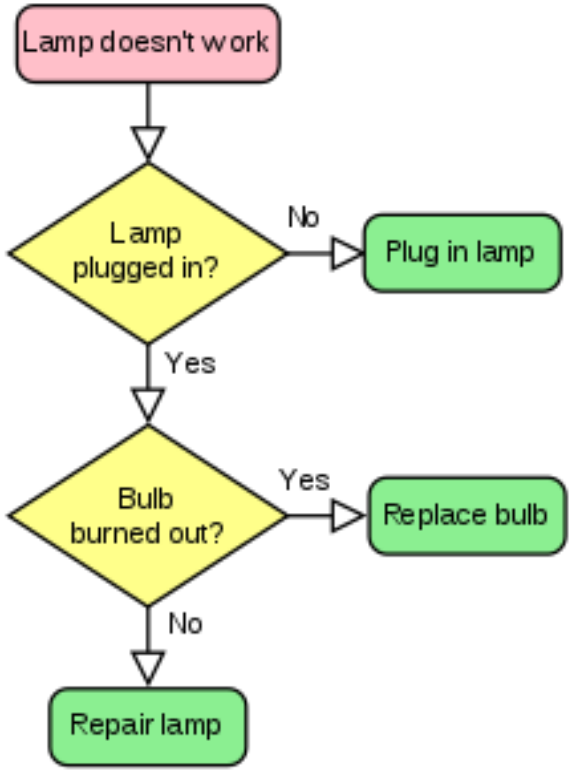
Exercise

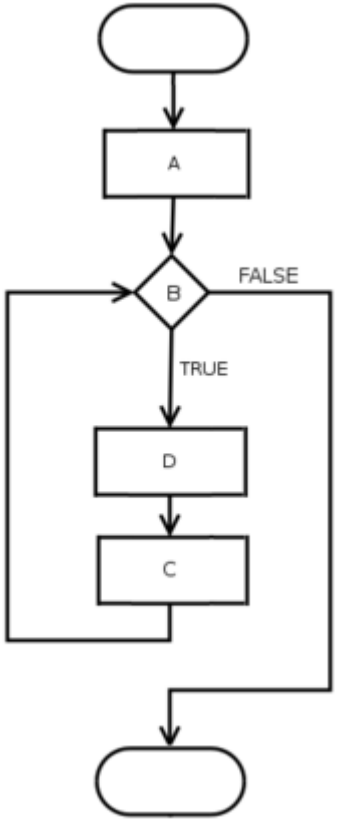
- A flowchart can be used to define an algorithm. The flowchart shows the steps as boxes (called node) of various kinds, and their order by connecting the boxes with arrows. Below are two different flowchart examples:





- Draw a class diagram that visualizes the relationships between a flowchart, nodes and arrows






```
class FlowChart {
  Node[] owns;
  Arrow[] has;
}
```

```
class Node {
  FlowChart partOf;

  String label;
  int type;

  Arrow[] outgoing;
  Arrow[] incoming;
}
```

```
class Arrow {
  FlowChart partOf;

  String label;

  Node startsAt;
  Node endsAt;
}
```

Answer