UML

The Basics of the Unified Modeling Language

The Basics of the Unified Modeling Language

- What is UML?
- UML Diagrams
- Review of Class Diagrams
- Tools and Resources

What is UML?

- UML stands for Unified Modeling Language
  - **Unified**: It brings together several techniques and notations for design, and has been standardized (first by the Object Management Group in 1997 and then by the International Organization for Standardization in 2005)
  - **Modeling**: It describes a software system and its design at a high level of abstraction
  - **Language**: It provides the means to communicate this design in a logical, consistent, and comprehensible fashion

What is UML?

- Goals of UML
  - Enable the modeling of object-oriented designs
  - Visually depict various aspects of the overall design of a solution
  - Provide extensibility and specialization mechanisms to extend core concepts
  - Be independent of particular programming languages and development processes
  - Support higher-level development concepts such as collaborations, frameworks, patterns, and components

What is UML?

- Brief history of UML
  - Object-oriented modeling languages began appearing between mid-1970 and the late 1980s
  - By the mid-1990s, there were dozens of options and variants; the space was cluttered and confusing to developers
  - It became necessary to create a unified approach with general acceptance to allow things to move forward
  - These efforts ultimately produced UML
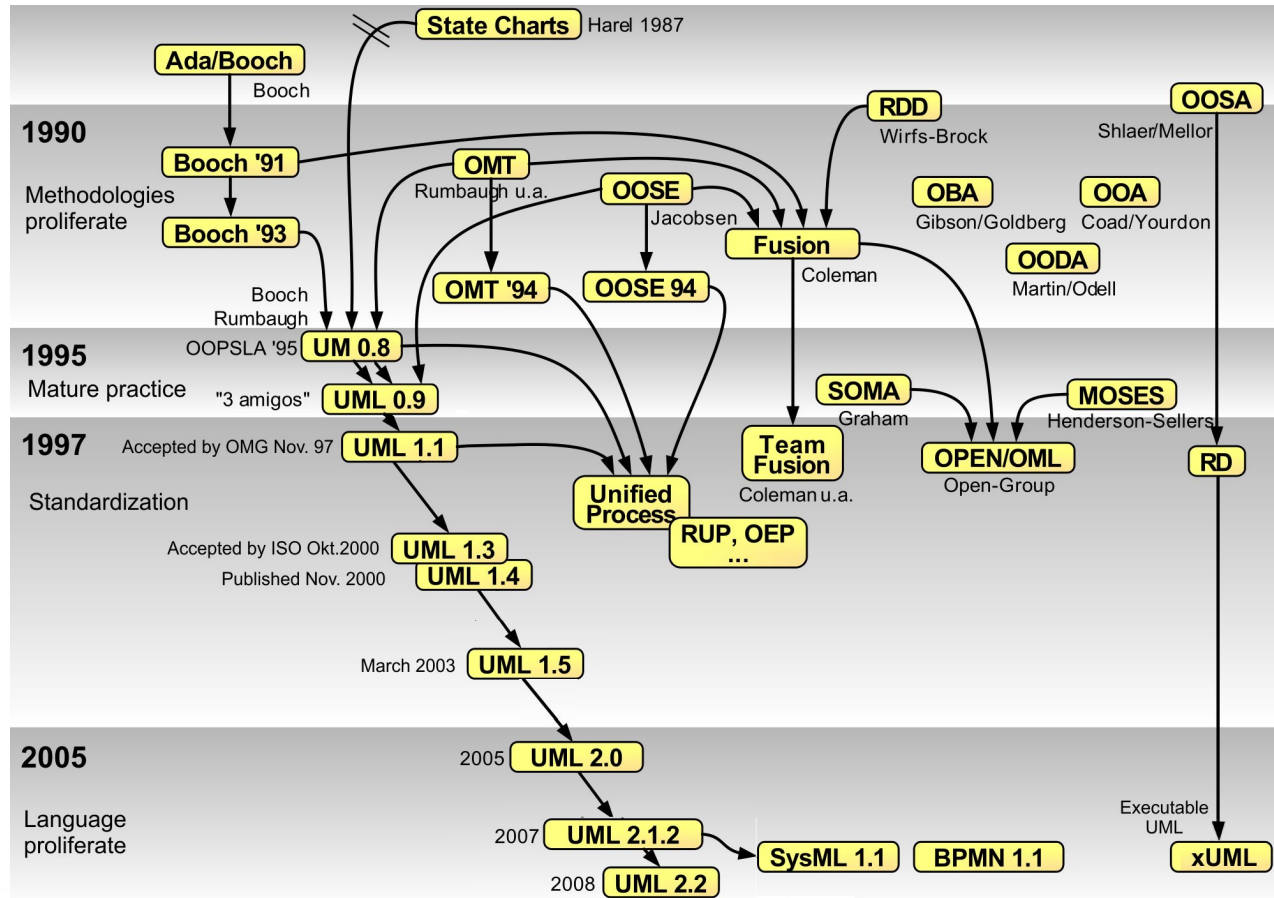
UML Diagrams

- Each UML diagram allows developers and customers to view a software system from different perspectives and from different levels of abstraction

- Some examples:
  - Structure diagrams (class, component, object, …)
  - Behaviour diagrams (use case, activity, …)
 - Interaction diagrams (sequence, timing, …)

UML Diagrams
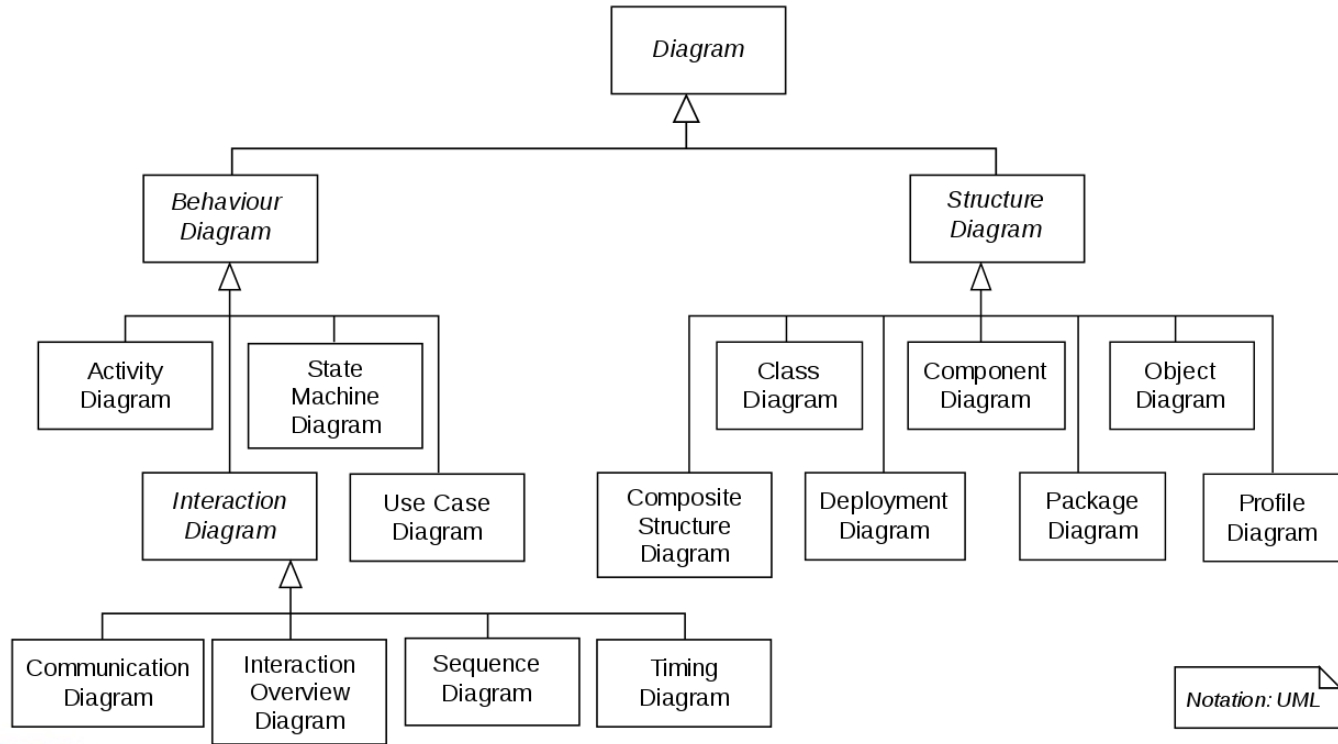
Review of Class Diagrams

- Class diagrams are widely used to describe the types of objects in a software system, their relationships, and constraints on their relationships

- In early stages of development, class diagrams might be incomplete, but are then refined and filled out as development proceeds
    - Some classes might be missing
    - Classes might be missing optional elements or details

- This is particularly the case in agile development …

Review of Class Diagrams

- Each class in a class diagram is represented by a rectangle divided into three sections:
  - **Name** of the class
  - **Attributes** of the class (the data members of the class, including both variables and constants)
  - **Operations** of the class (equivalent to member functions/methods)

Review of Class Diagrams

Person
firstName lastName
getFirstName() getLastName() setName( ) equals( ) toJSON()
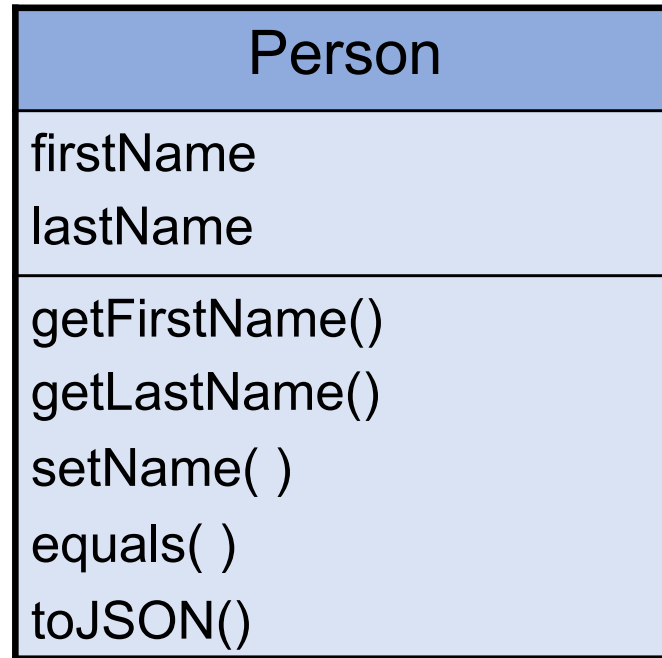
Review of Class Diagrams

- Attributes and operations may include:
  - Visibility: public (+), protected (#), or private (-)
  - Type of attribute or operation
  - Multiplicity of attribute (how many of this are there)
  - Parameter list for operations
  - Other properties of attributes or operations

- Including this information is of the form:
  - visibility variable_name: type
  - visibility variable_name: type multiplicity = default_value {property}
 - visibility method_name(parameter_list): return_type {property}

Review of Class Diagrams

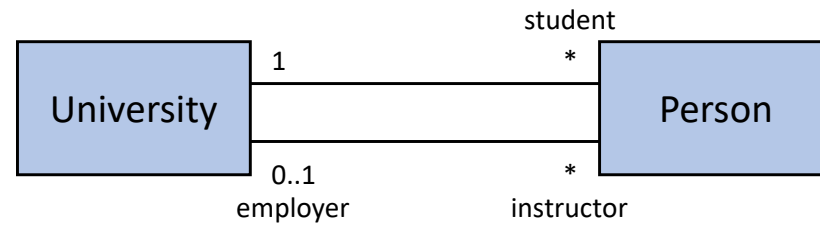| Person |
| --- |
| - firstName: string
- lastName: string |
| + getFirstName(): string
+ getLastName(): string
+ setName(first: string, last: string)
+ equals(otherPerson: Person): boolean
+ toJSON(): string |

Review of Class Diagrams

- **Associations** represent relationships between classes
  - Indicated with a solid line between the classes
  - Arrowheads are used to indicate the direction of the association; both unidirectional and bidirectional associations are possible
  - Can be annotated with a name for the association or roles that the classes play in the association
  - Can also include cardinalities to include the number of entities that can be involved with in the association, including one-to-one, one-to-many (1..*), many-to-many (*..*), zero-to-many (0..*), and so on

Review of Class Diagrams

| Person | 0..1 ———————— * | Car |

| Person | owner <—————————> 0..1 * | Car |

| Person | owns ▶ 0..1 * | Car |
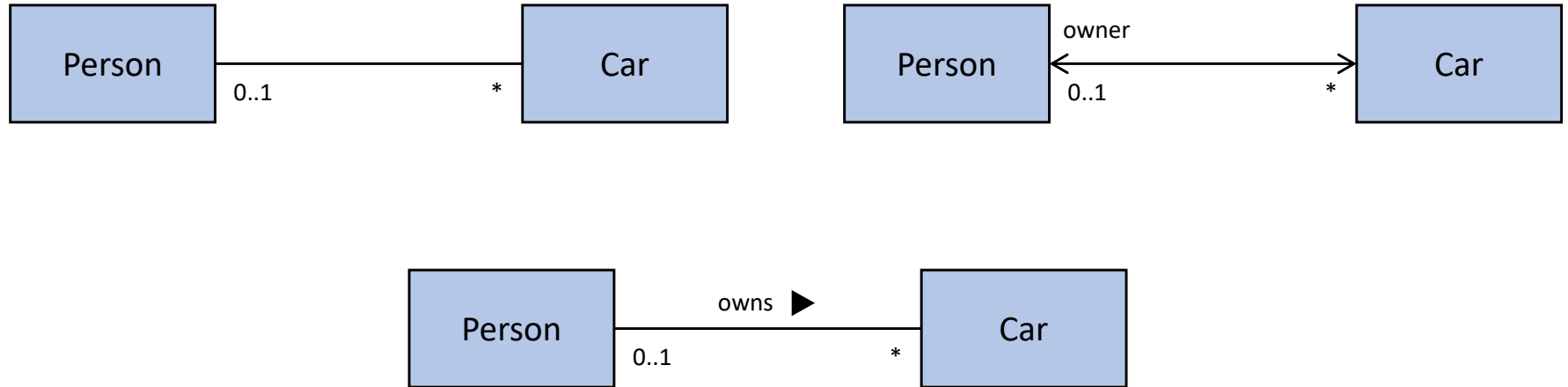
Review of Class Diagrams

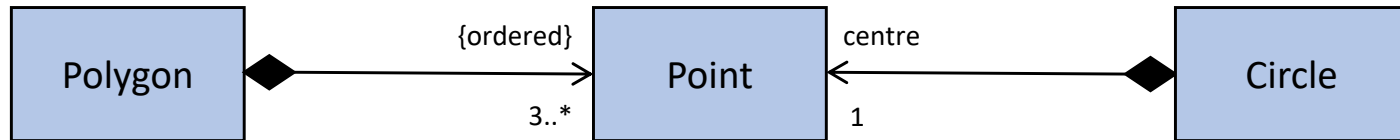# Review of Class Diagrams

- Two special forms of association are **aggregation** and **composition**
  - Aggregation (indicated by a hollow diamond) indicates a part-of relationship, but such parts are non-essential and can exist independent of the aggregate
  - Composition (indicated by a filled diamond) indicates that something is an essential and integral component of something else
    - Instances of a component may only have a single owner; sharing is not allowed
    - Components cannot exist independent of their owner
    - Components live or die with their owner

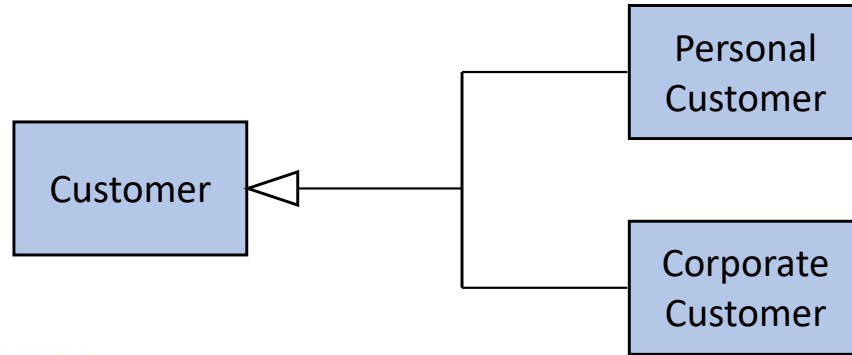# Review of Class Diagrams

Aggregation:
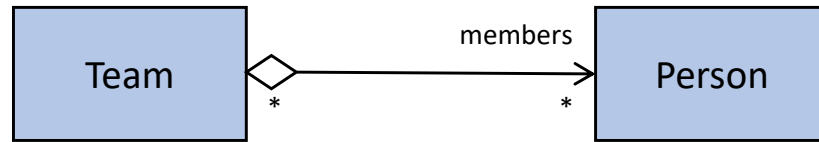
Composition:

Review of Class Diagrams

- **Generalization** is another form of relationship supported by UML
  - This is used to capture the object-oriented concept of inheritance
  - In other words, it allows you to indicate that one class is derived from another class, creating parent-child relationships between classes

Review of Class Diagrams

- A **dependency** between two classes exists if changes to one class might induce changes to or otherwise impact the other class
  - There are many types of dependency, including use, calling, creation, and so on
  - The broken line used to indicate the dependency can be labelled with a message to provide details on the dependency

Library Customer --- searches online catalogue ---> Computer
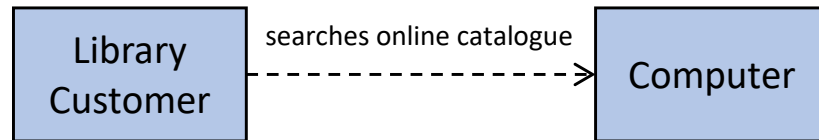
Review of Class Diagrams

- UML class diagrams can also contain many other general programming or object-oriented concepts and constructs
  - Notes and comments
  - Enumerations
  - Interfaces and abstract classes
  - Templates
  - Qualified associations
  - Association classes

Tools and Resources

- Many software tools exist to assist in the construction of the various types of UML diagrams
  - Microsoft Visio
  - Lucidchart
  - draw.io
  - …
- For a reasonably decent list of UML tools, check out:
  - https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools

Tools and Resources

- UML Distilled (now in its third edition) by Martin Fowler is an excellent resource for all things UML

- You can also get some good information on UML straight from the OMG at uml.org

- Time permitting, we will come back to look at some of UML's other diagrams later in the course …