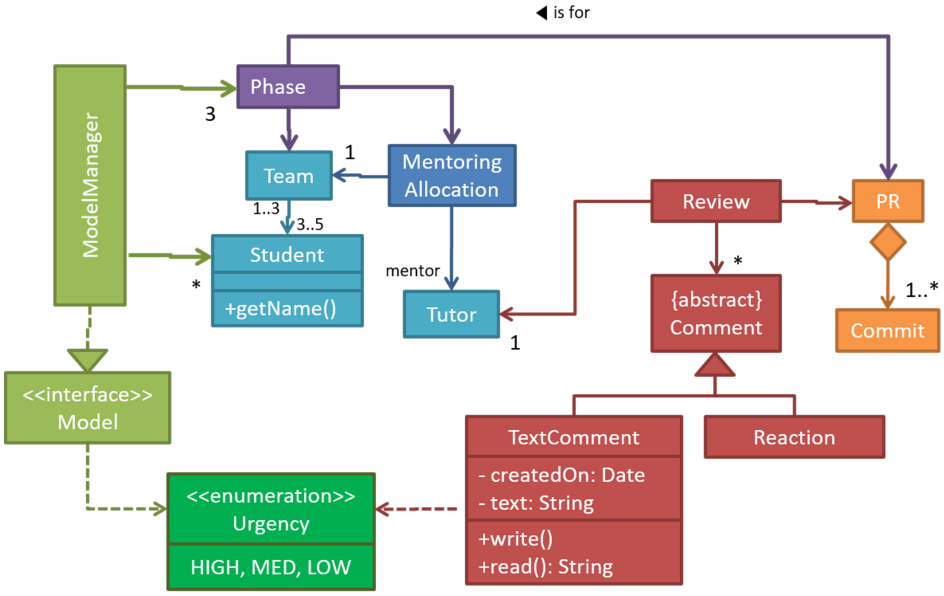
Unified Modelling Language (UML)

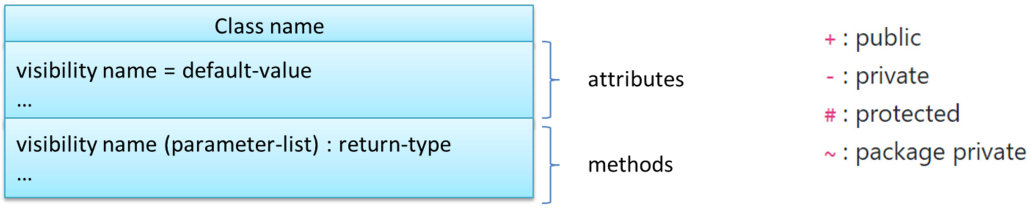
Friday, 21 September 2018 1:22 PM

- What is the relationship between all the diagrams? (class, object, sequence)

Class diagram

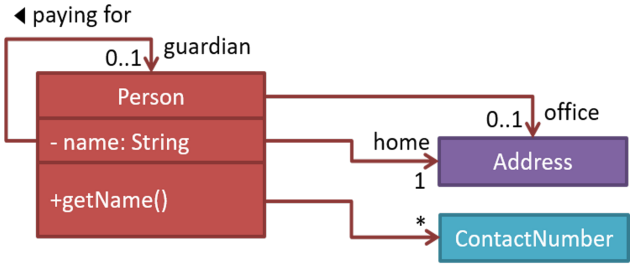


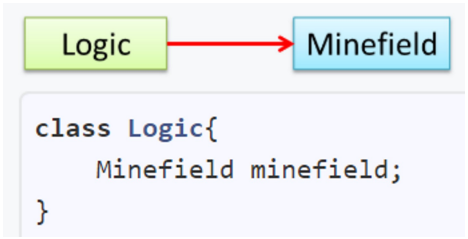
- Describes structure of OOP solution



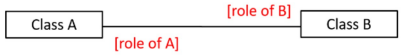
- Static (class-level) - underlined

Associations



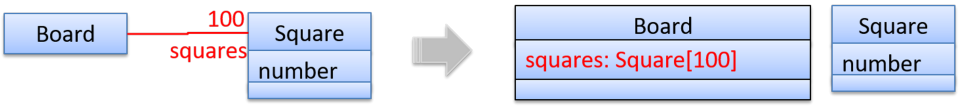


- Represented using solid line
- Navigability: use arrow
  - Bi-directional: difficult to maintain (coupling)
- Role

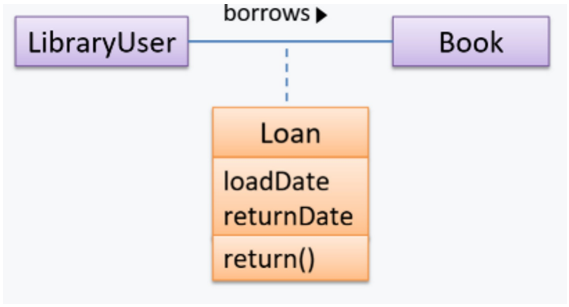


  - Usually matches with variable name in other class (e.g. Man class has wife variable)
- Label: meaning of association, use arrow to indicate direction to be read
- Multiplicity:
  - Multiplicity of A = how many objects of class A are associated with one object of class B
  - 0..1: optional
  - 1: compulsory
  - *: 0 or more

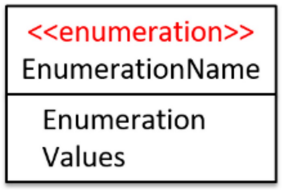
    - n..m: n to m inclusive
- Dependency: other object not kept as parameter, but used in method
    - Dashed arrow (e.g. on ENUMS)
- Association as an attribute



- Association classes
    - Store information about association
    - Dotted line from association



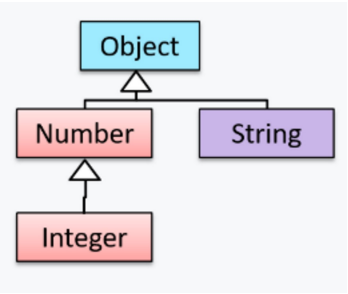
Enumerations



Composition vs Aggregation

- Composition = whole-part (e.g. Board-Square), has-a
    - Part cannot exist without whole
    - Used to indicate non-cyclics (folder, sub-folder)
    - Auto-deletion does not imply composition
    - Solid diamond
- Aggregation = container-contained (e.g. Club-Person)
    - Hollow diamond (omit altogether, only adds confusion)

Inheritance

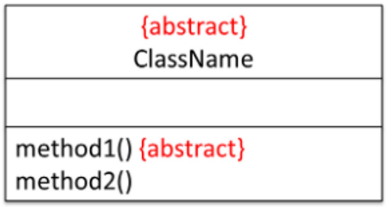


Interface

- `<<interface>> stereotype`
- Implement interface: same as inheritance, but with dotted lines

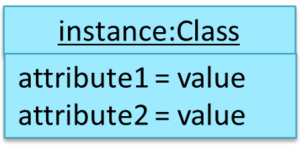
Abstract Classes and Methods

- `{abstract} keyword (or italics)`



Object Diagram

- Object structure at a given point of time
    - objectIdentifier :className
    - Properties = values (no methods)



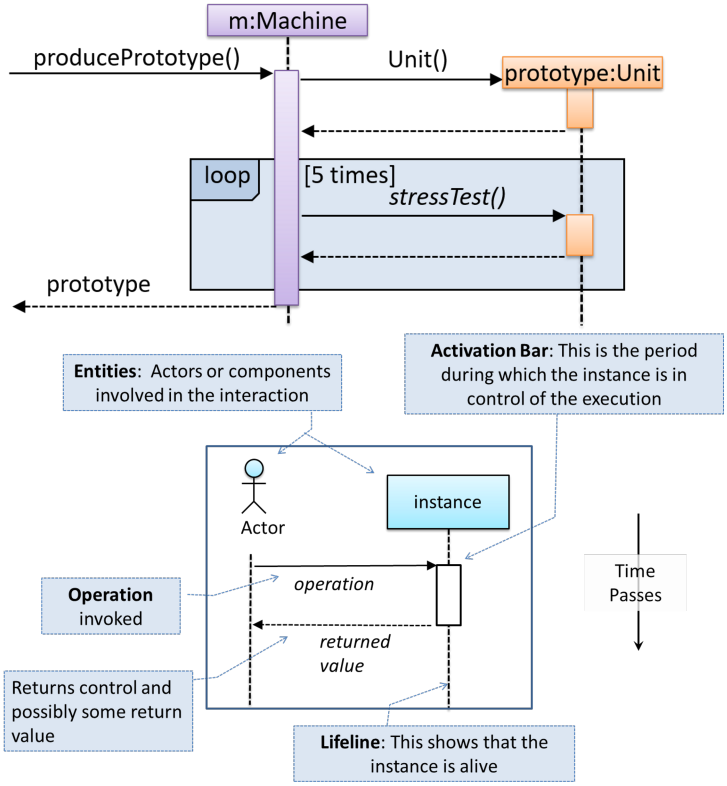
- Association = solid line, multiplicity omitted

Object vs Class Diagrams

- Instances (name, underline, :)
- No methods or multiplicities
- Multiple object diagrams correspond to one class diagram

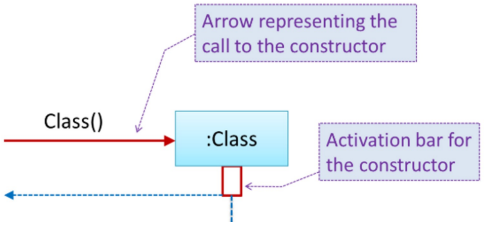
Sequence Diagram

- Captures interactions between multiple objects for a given scenario
  - Behaviour diagram (supported by other structure diagrams)
  - No underlining of classes/ objects here



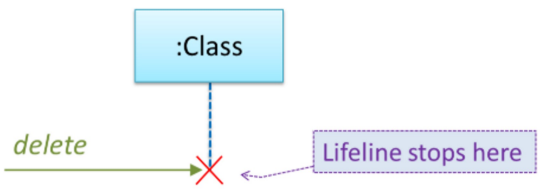
- Time moves downwards, Vertical lines = life lines
- Solid = call, dotted = return
  - Call: include parameters
  - Return arrows can be omitted if not ambiguous (e.g. Void)
  - Return variables are optional
- Activation bar: component doing something, ensure length is correct
  - Can be omitted if obvious (minimal notation, e.g. Constructors)

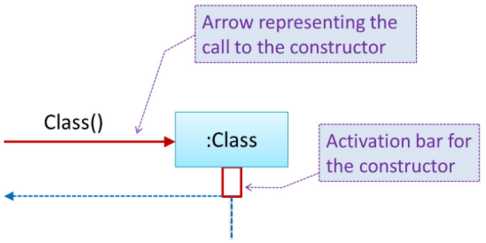
Object Creation

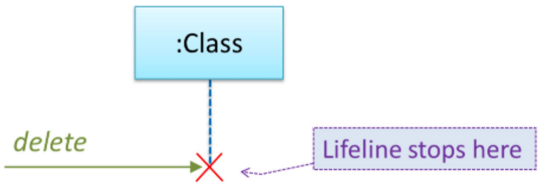


Object Deletion

- Destroyed using X at the end of lifeline



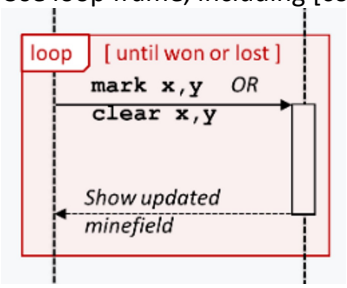




- Constructor method name = optional
- Instances can be unnamed

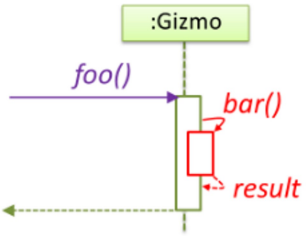
Loops

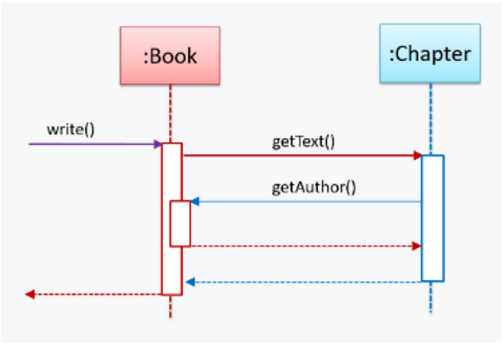
- Use loop frame, including [condition]



Self-Invocation

- Object calls its own methods

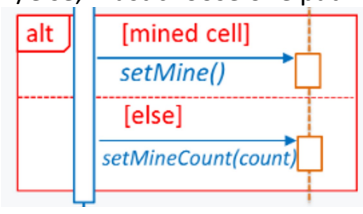




- Double activation box (have to call and return to itself)
- Call-back: A calls B, which requires calling A

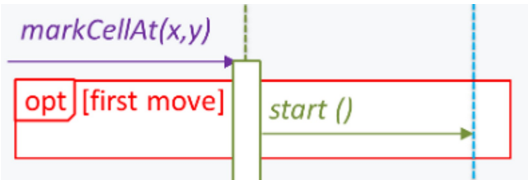
Alternative Paths

- [alt] frame
- if/else, must choose one path



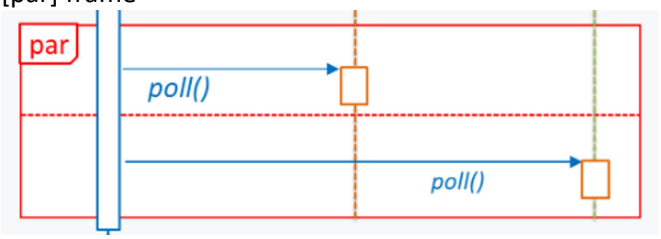
Optional Paths

- [opt] frame
- Lifeline must be within frame



Parallel Paths

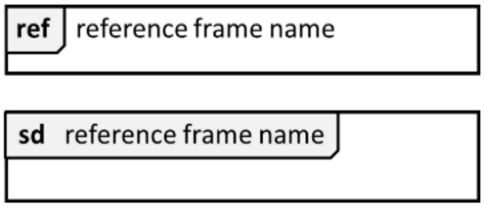
- [par] frame



- Java: implies multi-threaded program

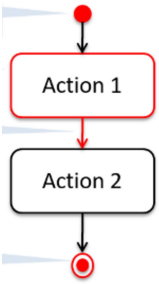
Reference Frames

- [ref] frame
- Segment of interaction shown in separate diagram



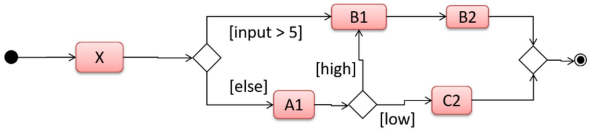
Activity Diagram

- Record workflow and control flows
- Action = rounded rectangle
- Control flow = solid line with arrow head



Alternate Paths

- Include branch node and merge node
  - Branch nodes should have guard condition (true = take path)
  - Only one guard condition can be true
  - Branches can join other branches

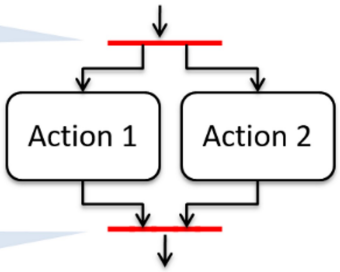


Parallel Paths

- Current flows on control
- Fork and join nodes enclose a set of parallel paths

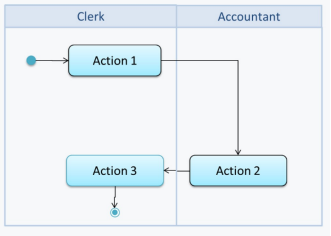
Fork (denotes the start of parallel paths - many outgoing edges)

Join (denotes the end of parallel paths – many incoming edges)



Rakes

- Separate activity diagram
- Activity: Move piece (new diagram)



Swimlanes

- Partition activity diagram to show who is doing which action
  - Swimlane diagrams

Notes

- Optionally connected to an element in the diagram
- `{constraint}`
  - Can be given in Object Constraint Language (OCL) or natural language