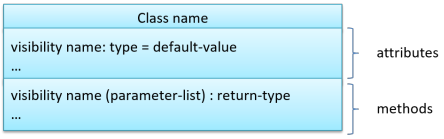
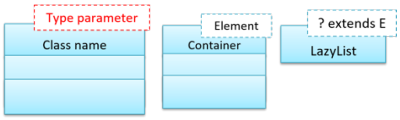
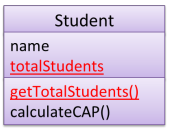
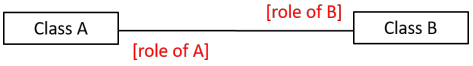
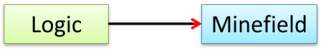
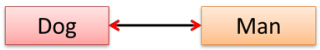
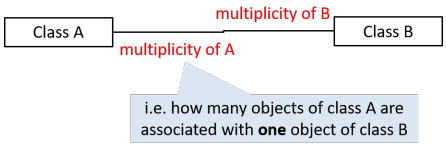
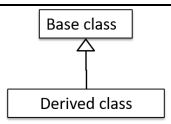
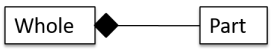
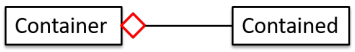
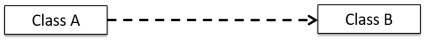
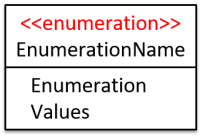
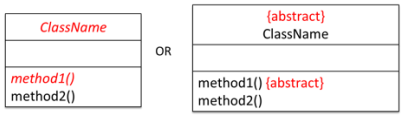
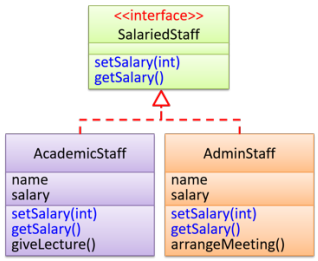
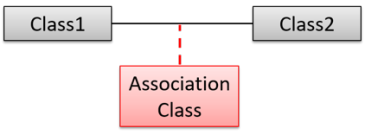
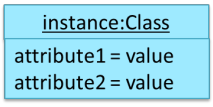
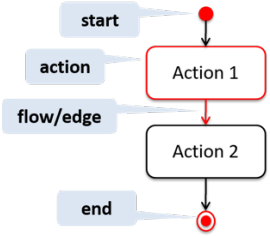
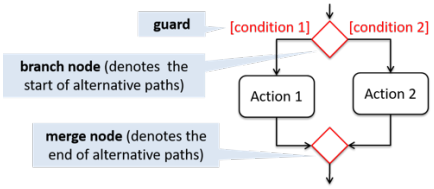
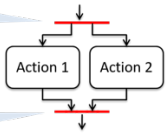
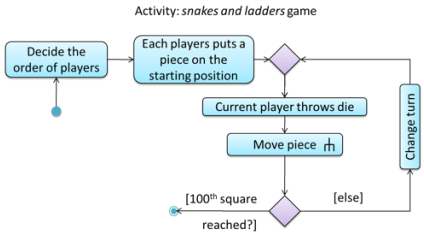
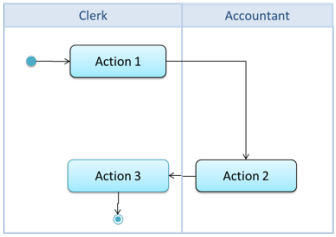
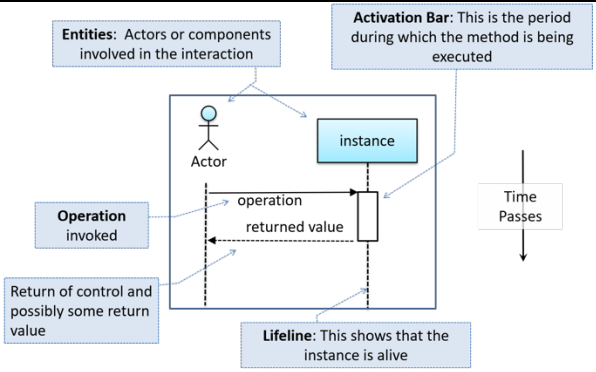
UML Diagram Notation

	Example
Class	
visibility	+ : public - : private # : protected ~ : package private
Generic	
static (Class Level)	Use an underline to indicate static 
Association	Use a solid line to show an association 
Association labels [Optional]	Describe the meaning of the association [Arrow head indicates the direction in which the label is to be read] 
Association Role [Optional]	
Navigability	Logic has ref. to Minefield  Dog and Man have ref. to each other 
Multiplicity	 Commonly used multiplicities: • 0..1 : **optional**, can be linked to 0 or 1 objects. • 1 : **compulsory**, must be linked to one object at all times. • * : can be linked to 0 or more objects. • n..m : the number of linked objects must be within n to m inclusive e.g., 2..5, 1..* (one or more), *..5 (up to five)
Inheritance	Does not matter triangle is filled or empty.

	
Composition	Use a solid diamond symbol 
Aggregation	Uses a hollow diamond 
dependencies	
Enumeration	
Abstract	*italics* or {abstract} (preferred) 
Interface and Implement	
Association class	
Object	 • class name and object name are underlined • no compartment for methods • Attributes compartment can be omitted • Object name can be omitted
Activity Diagram	
Alternate Paths [AD]	

	Some acceptable simplifications • Omitting the merge node if it doesn't cause any ambiguities. • Multiple arrows can start from the same corner of a branch node. • Omitting the [Else] condition.
Parallel Paths [AD]	**Fork** (denotes the start of parallel paths - many outgoing edges) **Join** (denotes the end of parallel paths – many incoming edges) 
rake notation [AD]	Indicate a part of activity is describe separately. 
swimlane diagrams [AD]	Show who is doing which action 

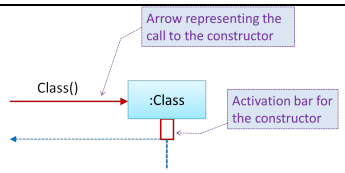
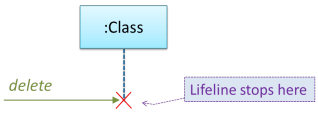
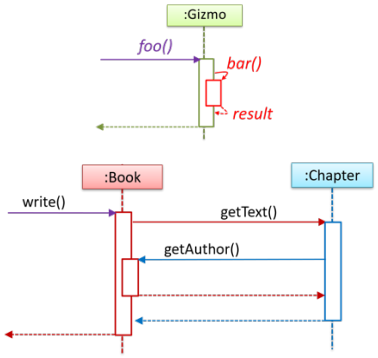
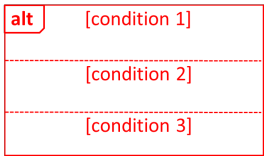
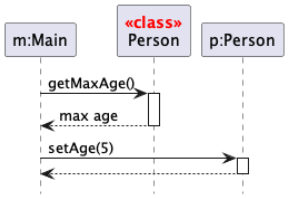
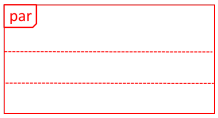


- Arrows representing method calls should be solid arrows while those representing method returns should be dashed arrows.
- Class/object name is not underlined in sequence diagrams.
- Arrows showing **synchronous** (i.e., the caller method is blocked from doing anything else until the called method returns) should use filled arrowheads (e.g., →).
- **Asynchronous** method calls (i.e., the caller method does not have to wait till the called method returns) are shown using lined arrowheads (e.g., →).

Optional elements

- activation bars
- return arrows
- method parameters use [foo(...) instead of foo(int size)]

Loops	loop [condition]

Object Creation	 • The arrow that represents the constructor arrives at the side of the box representing the instance. • The activation bar represents the period the constructor is active.
Object Deletion	Uses an X at the end of lifeline of an object to • show its deletion • indicate the point at which the object becomes ready to be **garbage-collected** 
Self-invocation	
alternative paths	 • No more than one altern. be executed • Acceptable for none to be executed
optional paths.	
static in [SD]	
parallel paths	
Reference Frames	Allow a segment of the interaction to be omitted and shown as a separate sequence diagram. - break complicated diagrams into multiple parts - omit details not interested in showing ref reference frame name sd reference frame name