UML FOR OOP DESIGN

UNIFIED MODELLING LANGUAGE

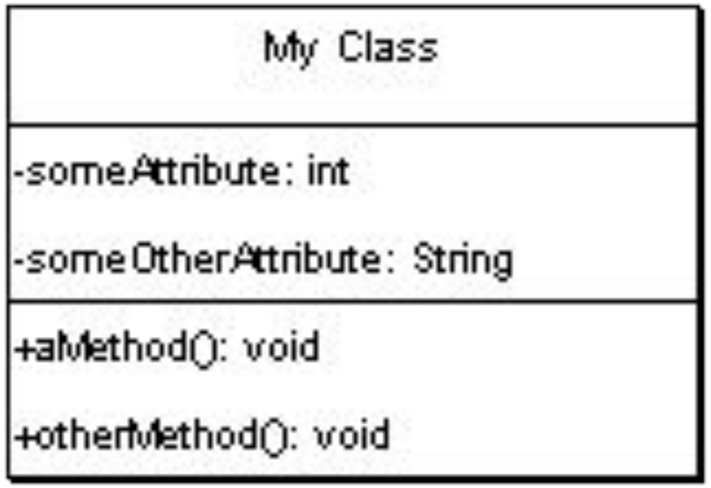
- UML is a diagramming tool for describing and documenting object oriented applications
- Programming language independent
- Used for modelling an application before its engineered
- Twelve different diagrams in all, with many complex details
- Generally though only two of these are used regularly
  - Class diagrams
  - Sequence diagrams

UNIFIED MODELLING LANGUAGE

- Class Diagrams
  - Describe classes and interfaces
  - …their properties
  - …their public interface
  - …and their relationships (e.g. inheritance, aggregation)
- Sequence Diagrams
  - Describe how objects send messages to one another
  - Useful for describing how a particular part of an application works
- We'll be covering just class diagrams
  - Very useful for describing APIs and discussing OO applications

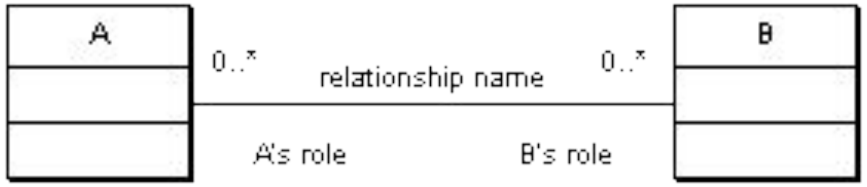
UML -- CLASSES

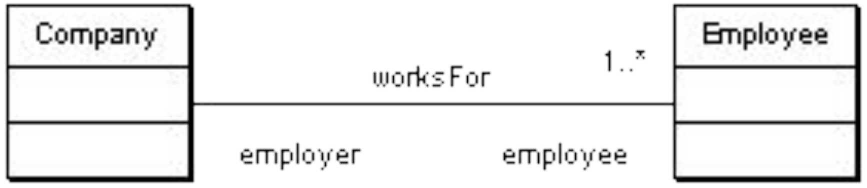
- Box with 3 sections
- The top contains the class name
- The middle lists the classes attributes
- The bottom lists the classes methods
- Can indicate parameters and return types to methods, as well as their visibility



UML -- ASSOCIATION

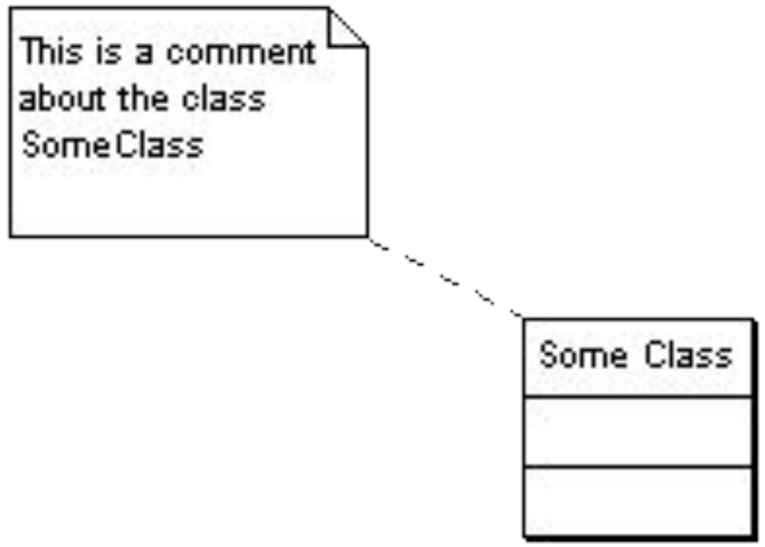
- A line between two classes indicates a relationship
- Extra information can be added to describe the relationship
- Including
  - Its name
  - The roles that the classes play
  - The *cardinality* of the relationship (how many objects are involved)
- E.g. a Person worksFor a Company, which has many employees





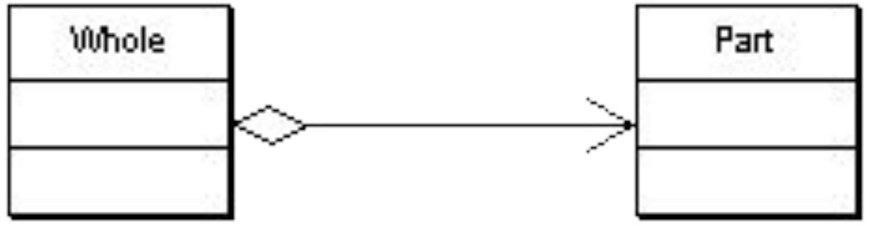
UML -- COMMENTS

- Useful for adding text for the readers of your diagram
- The symbol looks like a little post-it note, with a dotted line joining it to the class or relationship that its describing



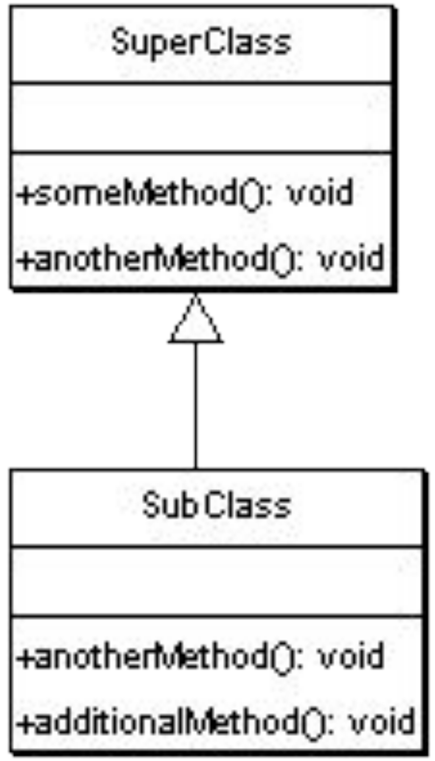
UML -- AGGREGATION

- Aggregation (a whole-part relationship) is shown by a line with clear diamond.
- As aggregation is a form of relationship you can also add the usual extra information
- I.e.
  - Name
  - Roles
  - Cardinality



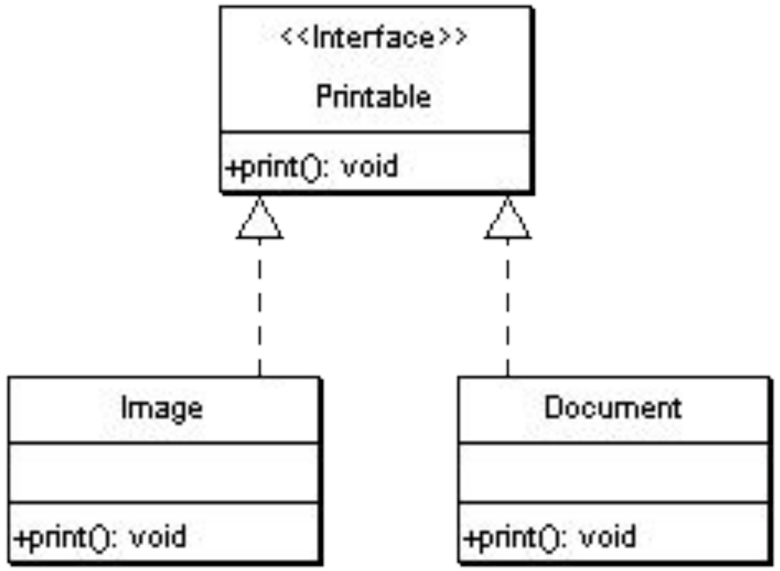
UML -- INHERITANCE

- Inheritance is shown by a solid arrow from the sub-class to the super-class
- The sub-class doesn't list its super-class attributes or methods,
  - *unless* its providing its own alternate version (i.e. is extending the behaviour of the base class)



UML -- INTERFACES

- Interfaces are a way to specify behaviour (a public contract) without data or implementation.
- Interfaces are classed with an extra label next to their name: `<<Interface>>`
- A dotted arrow from a class to an interface explains that the class fulfills the contract specified by that interface


<<Interface>>
Printable
+print(): void
Image
+print(): void
Document
+print(): void