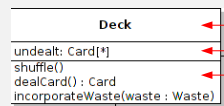


UML Notation for Class Diagrams

Document the resulting structure with a *class diagram* showing

- class, method, and parameter names
- parameter and return value types
- key instance variables
- relationships between classes



class name
instance variables
methods

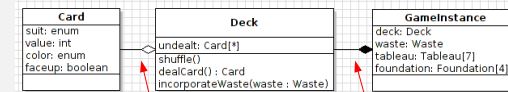
[*] denotes "some variable number" i.e. a collection
it may or may not be implemented with an array

additional elements can be added
e.g. C or I to denote class or interface, or symbols to denote private/public/protected elements

UML Notation for Class Diagrams

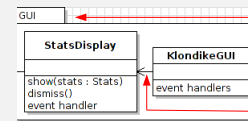
Showing relationships between classes –

- instance variables → aggregation or composition
- other things (e.g. parameters) → association



aggregation

composition



module (package)

association for collaborations where aggregation and composition do not apply

Aggregation vs. Composition



aggregation

composition

Aggregation:

- a part-whole relationship where the whole does not control the lifecycle of the part
- the part may be created before and/or last longer than the whole
- the part may be accessed without going through the whole

Composition:

- a part-whole relationship where the whole controls the lifecycle of the part
- whole creates and destroys the part
- the part is not accessed without going through the whole

Aggregation vs. Composition

Aggregation:

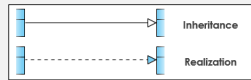
- part can exist independently of the whole

Composition:

- part cannot exist independently of the whole

- house and rooms
 - composition
- class and students
 - aggregation
- a bag of marbles
 - aggregation
- car and engine
 - how things are modeled can depend on the needs of the program
 - the engine can exist separately from the car, but it typically doesn't and it may be more appropriate to treat the engine as a component

UML Notation for Class Diagrams



inheritance for subclasses (extends)
realization for implementing interfaces (implements)