Class Relationship Summary

Relationship / Attributes	Inheritance (aka Generalization)	Association	Aggregation	Composition	Dependency (aka using, delegation)
Semantics (meaning)	• One object is *a kind* of another object • "Is a" relationship ("a car is a vehicle") • One object inherits attributes & operations from another	• *Parts of* relationship • "Has a" relationship ("a class has a teacher") • Objects are peers	• *Parts of* relationship • "Has a" relationship ("a car has a battery") • Build a complex whole object from simple part objects	• *Parts of* relationship • "Has a" relationship ("a car has an engine") • Build a complex whole object from simple part objects	• One object *depends* on another object • On object *uses* the services of another • An object *delegates* some responsibility to another
Navigation / Knowledge	Unidirectional (Child to Parent)	Bidirectional	Unidirectional (Whole to Part)	Unidirectional (Whole to Part)	Unidirectional (Dependent to Independent)
Object Binding	Strong	Weak	Weak	Strong	Temporary
Persistence	Permanent	Dynamic	Dynamic	Permanent	Transient
Object Lifetimes	Simultaneous	Independent	Independent	Simultaneous	Different
Object Sharing	Sub-objects instantiated from parent class are embedded in a single child-class object	An object may be associated with many objects	Parts may belong to multiple wholes	Parts belong to one whole	Independent may be used by multiple dependents
Implementation	Keyword / special syntax (not used for other purposes)	Class scope variable in both classes	Class scope variable in whole class only	Class scope variable in whole class only	Local variable or argument in dependent class function or method
C++	`: public`	Pointer variable	Pointer variable	Object variable (static instantiation)	Pointer or object
Java	`extends`	Reference variable	Reference variable	final reference variable	Reference variable
UML representation					

Class Relationships

1. Inheritance (also known as Generalization)
2. Association
3. Aggregation
4. Aggregation
5. Dependency (also known as using or delegation)

Critical Attributes and Legal Values

Attribute	Legal Values	Meaning
Semantics [1]	Kind of	One object is a kind of another object (“IS A”)
	Part of	One object is a part of another object (“IS A PART OF” or “HAS A” or “CONTAINS A”)
	Depends on	One object depends on (uses, delegates some responsibility to) another object
Navigation[2]	Bidirectional	Possible to go from either object to the other (objects "know" about each other)
	Unidirectional	Possible to go from one object to the other but not in the opposite direction (only one object "knows" about the other)
Binding Strength[3]	Strong	• Object lifetimes are the same: they are created and destroyed at the same time • Relationship is static: once the relationship between two objects is formed, it cannot be changed or broken
	Weak	• Object lifetimes <u>may</u> be different: they <u>may</u> be created and destroyed at different times • Relationship is dynamic: can last indefinitely but may be changed or broken when needed
	Temporary	• Object lifetimes are different: they <u>must</u> be created and destroyed at different times • Relationship is transient: it is created at the beginning of a function or method call and ends when the function or method completes

1. What does the relationship mean in the problem domain.
2. Given one object in the relationship, is it possible to navigate to or access the other object.
3. Binding strength summarizes two attributes: The lifetimes of related objects are the same if they are created and destroyed at the same time and they are different if they may be created and/or destroyed at different times. Furthermore, the relationships between some objects are established when the objects are created and persist as long as the objects exist, while other relationships are more ephemeral and are established and terminated more dynamically.