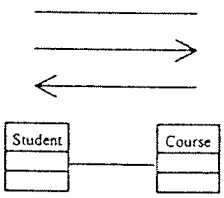
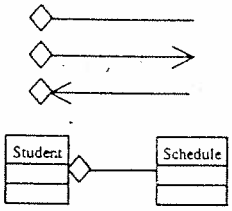
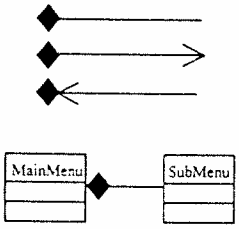
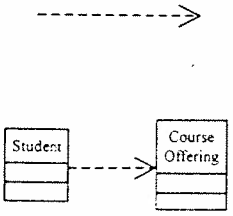
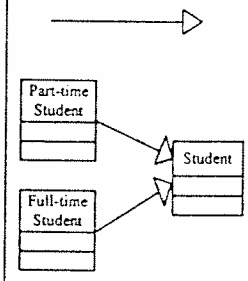
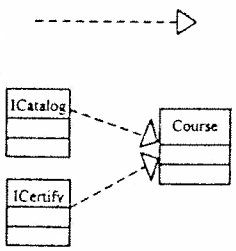
Class Relationship Chart (with examples)

Relationship	UML Notation	Test Phrase	Characteristics	Notes
Association		"knows a"	▪ Structural ▪ Semantic ▪ Permanent / Persistent ▪ Object Motivated	Typically used with classes but is also the actor to use case relationship which is stereotyped to <<communicates>>. Remember the default class relationship during analysis is a bi-directional association (arrowless line)
Aggregation		"has a" "is a part of" "contains"	▪ Structural + ▪ Semantic + ▪ Permanent / Persistent ▪ Object Motivated	I call these aggravations sometimes as I see too much discussion about them during analysis offering little value there. If unsure then just hold off this decision until a closer look in design clears it up.
Composition		"is composed of" "owns"	▪ Structural ++ ▪ Semantic ++ ▪ Permanent / Persistent ▪ Object Motivated ▪ Coincidental Lifetimes	Remember that aggregations & compositions are kinds of associations. If making the call that ownership is implied you are telling the coder to make certain direction of communication is down to the part supporting encapsulation of the family of classes.
Dependency		"uses" "requires the service of"	▪ Non-Structural ▪ Semantic ▪ Temporary / Transient ▪ Class Motivated ▪ Client / Server	This is the "use it then lose it" relationship that analysis associations often refine to which loosens the coupling of the system by weakening the structural aspects of the system being built. The decision to go with a dependency is driven by how the relationship will be implemented to meet all scenario needs.
Generalization		"is a" "is a kind of" "is a special type of"	▪ Ultimate Structural as it models a single object / instance. ▪ Semantic ▪ Class and Reuse Motivated ▪ Polymorphism Implementer	This relationship can be thought of as a mold or cast you are using to shape the object for needs in a given context. Used with abstract base classes, it is the relationship which can be a direct, measurable opportunity for reuse and polymorphic behavior.
Realization		"realizes" "is realized by" "implements" "is implemented by"	▪ Non-Structural ▪ Class/Subsystem Motivated ▪ Interface ▪ Polymorphism Implementer	This realization models a special relationship between a class or subsystem and the public interface which holds the operation signatures which that and any other implementing class or subsystem must completely deliver. Also used with use case realizations.

Structural = visibility to member data and functions Semantic = application context drives relationship
Permanent = relationship outlives session Persistent = mirrored in the database Transient = gone after use