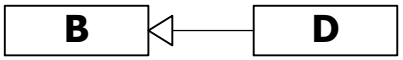
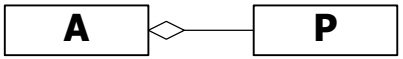
Relationships between C++ Classes

Relationship	Diagram	Code	Explanation
Inheritance D "is-a" B		class D : public B { … };	Derived class D is a specialization of the Base class B. D inherits all the members of B except constructors

Composition
 	
Ownership, P is "part-of" C
		class C {
		…
		Private:
		P p;
		};
			Composite class C owns, or contains, a part class P. P is created and destroyed with C. The interface of P is visible only to C, not its clients.

Aggregation
 	
Ownership, P is "part-of" A
		class A {
		…
		Void fun()
		{ P* ptrP = new P(); … }
		};
			The Aggregator class A owns a part class P. P is created by a member function of A, and so its lifetime is strictly less than that of A. A is expected to destroy P.

Using
 	
Referral:
U uses R through a reference
		public class U {
		…
		public void register(R& r)
		{
		// use r
		}
		};
			A class U uses instance of class R, to which it holds a reference. R is created by some other entity and a reference to it is passed to some member function of class U.

Relationships between C++ Classes