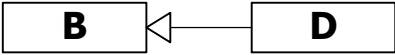

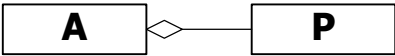
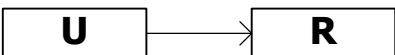


## Relationships between C++ Classes

Relationship	Diagram	Code	Explanation
<p>Inheritance</p> <p>D "is-a" B</p>		<pre>class D : public B { ... };</pre>	<p>Derived class D is a specialization of the Base class B. D inherits all the members of B</p>
<p>Composition</p> <p>Ownership, P is "part-of" C</p>		<pre>class C { ... Private: P p; };</pre>	<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.</p>
<p>Aggregation</p> <p>Ownership, P is "part-of" A</p>		<pre>class A { ... Void fun() { P* ptrP = new P(); ... } };</pre>	<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.</p>
<p>Using</p> <p>Referral: U uses R through a reference</p>		<pre>public class U { ... public void register(R&amp; r) { // use r } };</pre>	<p>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.</p>