Distinguishing Between Delegation, Composition And Aggregation

Delegation

```
1  public class A {
2    private B b = new B();
3
4    public void method() {
5      b.method();
6    }
7  }
```

When clients of A call method, **class** A delegates the method call to B.
Rationale. Class A can inherit from one class, but expose behaviours that belong elsewhere.
Further Study. **http://beust.com/java-delegation.html**

Composition

```
1  public class A {
2    private B b = new B();
3
4    public A() {
5    }
6  }
```

Once there are no more references to a particular instance of **class** A, its instance of **class** B is destroyed.
Rationale. Allows classes to define behaviours and attributes in a modular fashion.
Further Study. **http://www.artima.com/designtechniques/compoinh.html**

Aggregation

```
1  public class A {
2    private B b;
3
4    public A( B b ) {
5      this.b = b;
6    }
7  }
```

```
1  public class C {
2    private B b = new B();
3
4    public C() {
5      A a = new A( this.b );
6    }
7  }
```

Once there are no more references to a particular instance of **class** A, its instance of **class** B will not be destroyed. In this example, both A and C must be garbage collected before B will be destroyed.
Rationale. Allows instances to reuse objects.
Further Study. **http://faq.javaranch.com/java/AssociationVsAggregationVsComposition**

Demonstration Without References

The names given to these simple patterns are defined by their referential relationships.