Symptoms of Bad Code/Bad Design

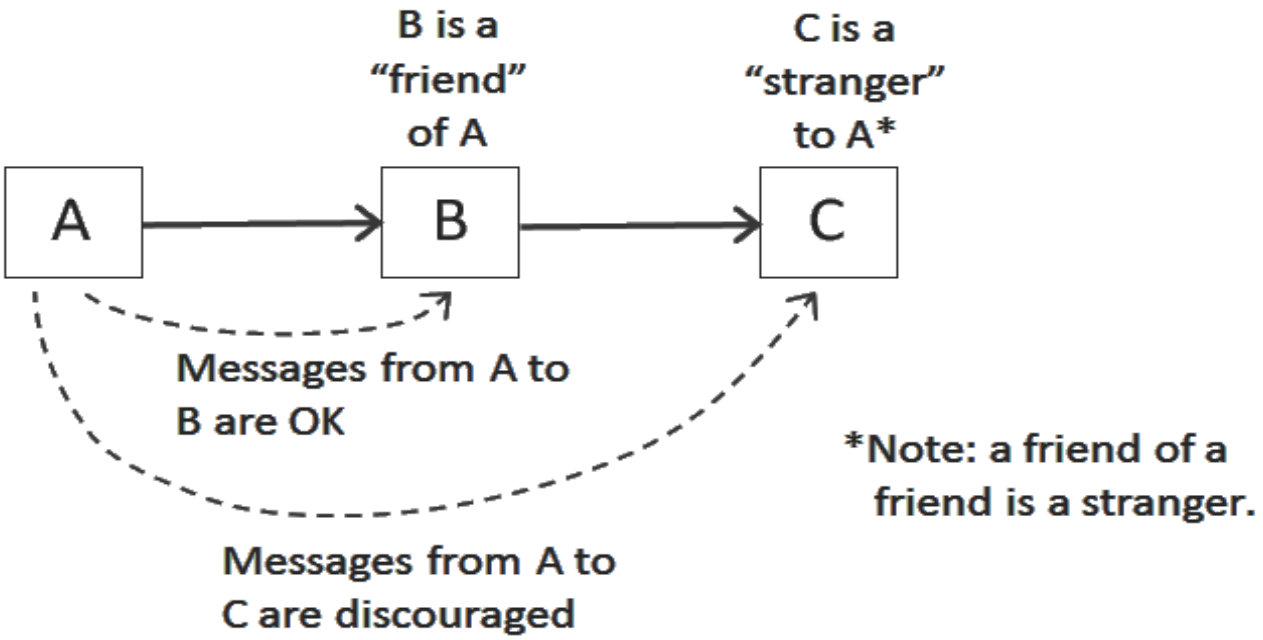
RIGIDITY	Difficult to change. Tightly coupled.
FRAGILITY	Breaks in many places when it is changed.
IMMOBILITY	Refers to inability to reuse code.
VISCOSITY	The resistance against making a change (long compile time, long deploy time etc.).

Object Oriented Design Principles (Uncle Bob SOLID)

SINGLE RESPONSIBILITY PRINCIPLE (SRP)	**A CLASS SHOULD HAVE ONLY ONE REASON TO CHANGE.**
OPEN/CLOSE PRINCIPLE (OCP)	**SOFTWARE ENTITIES (CLASSES, MODULES, FUNCTIONS ETC.) SHOULD BE OPEN FOR EXTENSION, BUT CLOSE FOR MODIFICATION.**
LISKOV SUBSTITUTION PRINCIPLE (LSP)	**FUNCTIONS THAT USE POINTERS OR REFERENCES TO BASE CLASSES MUST BE ABLE TO USE OBJECTS OF DERIVED CLASSES WITHOUT KNOWING IT.***
INTERFACE SEGREGATION PRINCIPLE (ISP)	**CLIENTS SHOULD NOT BE FORCED TO DEPEND UPON INTERFACES THAT THEY DO NOT USE.**
DEPENDENCY INVERSION PRINCIPLE (DIP)	**A.HIGH LEVEL MODULES SHOULD NOT DEPEND UPON LOW LEVEL MODULES. BOTH SHOULD DEPEND UPON ABSTRACTIONS.** **B.ABSTRACTIONS SHOULD NOT DEPEND UPON DETAILS. DETAILS SHOULD DEPEND UPON ABSTRACTIONS.**

*You should be able to replace an object with any of derived classes. Your code should never have to check which subtype it's dealing with.

Law of Demeter (LOD) - a way of decoupling



LOD VIOLATION: A.getThis().getThat().doTheWork()