**S O L I D**

Single Responsibility

Open-Close:

We worry about expectations of clients from the exposed abstraction(s). We try to be extensible to meet extra expectations of client(s) but we don’t modify the existing code because we might break some expectations.

Liskov:

We worry about run-times. We again worry about clients’ expectations from the exposed abstraction(s). In a hierarchical structure all derived types should be able to be used as base type without breaking the expectations. We control this principle in 5 areas:

. Pre-conditions: cannot be strengthened

. Post-conditions: cannot we weakened

. Invariants: must be preserved

. Exceptions: no new one should be thrown in derived types

. Contravariance of method arguments and Covariance of return types

All pre/post conditions are mech sometimes to protect invariants. All of them are categorized in Business rules. Therea are mainly 2 ways of handling business rules:

. Structural: one-2-one or one-2-many relationships are good examples

. Behavioral: all pre/post conditions

Interface segregation

Dependency inversion

**Why to use Builder?**

Because calling constructor is most the times complex (building IOptions requires many fields), then we prefer to make the constructor Internal and the construction is delegated to the Builder. So that’s why we use builder in production code.

**Why to use the Manager to update?**

According to the previous section (why to use builder) calling methos most of the times is a complex action, just like the constructors we prefer to make all methods Internal and then the invocations is delegated to the Manger class.

So, up to there we see than the Manager class can do both Constructing and Updating.

All these efforts, help us to avoid having double standards among the code.