Clean, Testable and Predictable methods

* Each method has a clear single purpose. Do 1 thing and do it well. This makes it easier to understand, easier to debug and easier to maintain and modify. Easier to write unit tests
* Has a good name that identifies its purpose.
* Focused code. The code itself makes the intent of the method clear. Reduces the need for extensive comments and should not have any unexpected side effects. Should not perform unrelated actions, change global data or modify params.
* Short length. So it’s clean and testable. Code that is long suggests not a single purpose
* Needs to be testable. Should be able to call it from an automated code test. Automated code tests increases quality and minimises bugs introduced by changes.
* Return a predictable result

Define names that are clear, conscise, specific accurate and unambiguous.

Unexpected side effects are actions taken by a method that it wasn’t designed for or hinted at in its name and single purpose.

Code-behind file – refers to code that sits behind an action or event. Difficult to test unless the logic is refactored into a separate method.

Pattern – a predefined way to build a set of code for a specific purpose.

Repository pattern contains a set of classes that handle the data access for your application. The repository class hides the details of accessing the data.

Flags:

Bool flags can be passed so their intent is more obvious:

Public void AMethod( bool theFlag){…}

AMethod(theFlag:true);