Five principles that should guide you every time you write code.

*SOLID* is the acronym for a set of best practices that, when imple­mented together, make code adaptive to change. The SOLID practices were introduced by Bob Martin. Even so, these practices are not as widely known as they could be—and perhaps should be.

Below are the SOLID principles:

**S** The single responsibility principle

**O** The open/closed principle

**L** The Liskov substitution principle

**I** Interface segregation

**D** Dependency injection

Even taken in isolation, each of these principles is a worthy prac­tice that any software developer would do well to learn. When used in collaboration, these patterns give code a completely dif­ferent structure—one that lends itself to change.

However, take note that these patterns and practices, just like all others, are merely tools for you to use. Deciding when and where to apply any pattern or practice is part of the art of soft­ware development. Overuse leads to code that is adaptive, but on too fine-grained a level to be appreciated or useful. Overuse also affects another key facet of code quality: readability. It is far more common for software to be developed in teams than as an individual pursuit. Thus, judiciously selecting when and where to apply each pattern, practice, or SOLID principle is imperative to ensure that the code remains comprehensible in the future.