**THE STRATEGY DESIGN PATTERN**

The GoF book says the Strategy design pattern should: “Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.”

The Strategy design pattern points out that, sometimes, it’s good to be taskoriented. That’s especially important if you want to maintain volatile code away from the main code for your app, or if you want to change the algorithm you use at runtime.

Consider using the Strategy design pattern if you have one of the following situations:

* You have volatile code that you can separate out of your application for easy maintenance.
* You want to avoid muddling how you handle a task by having to split implementation code over several inherited classes.
* You want to change the algorithm you use for a task at runtime.

So, there you have it — any time you start to get task-oriented and want to make those tasks one of the main design points of your code, the Strategy design pattern should spring to mind. That’s the way design patterns work.

They don’t provide you with specific code. Instead, you familiarize yourself with the idea, and when that idea could come in handy there’s an Aha! moment. This looks like a job for the Strategy pattern!