SUPER MARKET PRICING – Problem Statement

This kata arose from some discussions we’ve been having at the DFW [Practioners](http://www.pragmaticprogrammer.com/cgi-local/pragprog?DallasPractitioners) meetings. The problem domain is something seemingly simple: pricing goods at supermarkets.

Some things in supermarkets have simple prices: this can of beans costs $0.65. Other things have more complex prices. For example:

* three for a dollar (so what’s the price if I buy 4, or 5?)
* $1.99/pound (so what does 4 ounces cost?)
* buy two, get one free (so does the third item have a price?)

This kata involves no coding. The exercise is to experiment with various models for representing money and prices that are flexible enough to deal with these (and other) pricing schemes, and at the same time are generally usable (at the checkout, for stock management, order entry, and so on). Spend time considering issues such as:

* does fractional money exist?
* when (if ever) does rounding take place?
* how do you keep an audit trail of pricing decisions (and do you need to)?
* are costs and prices the same class of thing?
* if a shelf of 100 cans is priced using "buy two, get one free", how do you value the stock?

This is an ideal shower-time kata, but be careful. Some of the problems are more subtle than they first appear. I suggest that it might take a couple of weeks worth of showers to exhaust the main alternatives.

**Goal**

The goal of this kata is to practice a looser style of experimental modelling. Look for as many different ways of handling the issues as possible. Consider the various tradeoffs of each. What techniques use best for exploring these models? For recording them? How can you validate a model is reasonable?

**What’s a Code Kata?**

As a group, software developers don’t practice enough. Most of our learning takes place on the job, which means that most of our mistakes get made there as well. Other creative professions practice: artists carry a sketchpad, musicians play technical pieces, poets constantly rewrite works. In karate, where the aim is to learn to spar or fight, most of a student’s time is spent learning and refining basic moves. The more formal of these exercises are called kata.

To help developers get the same benefits from practicing, we’re putting together a series of code kata: simple, artificial exercises which let us experiment and learn without the pressure of a production environment. Our suggestions for doing the kata are:

* find a place and time where you won’t be interrupted
* focus on the essential elements of the kata
* remember to look for feedback for every major decision
* if it helps, keep a journal of your progress
* have discussion groups with other developers, but try to have completed the kata first

There are no right or wrong answers in these kata: the benefit comes from the process, not from the result.