>

CodeKata

Because experience is the *only* teacher

- RSS

Text

Search

Navigate…

- PragDave
- Kata
- Archives

Kata04: Data Munging

Martin Fowler gave me a hard time for Kata02, complaining that it was yet another single-function, academic exercise. Which, or course, it was. So this week let's mix things up a bit.

Here's an exercise in three parts to do with real world data. Try hard not to read ahead—do each part in turn.

Part One: Weather Data

In weather.dat you'll find daily weather data for Morristown, NJ for June 2002. Download this text file, then write a program to output the day number (column one) with the smallest temperature spread (the maximum temperature is the second column, the minimum the third column).

Part Two: Soccer League Table

The file football.dat contains the results from the English Premier League for 2001/2. The columns labeled 'F' and 'A' contain the total number of goals scored for and against each team in that season (so Arsenal scored 79 goals against opponents, and had 36 goals scored against them). Write a program to print the name of the team with the smallest difference in 'for' and 'against' goals.

Part Three: DRY Fusion

Take the two programs written previously and factor out as much common code as possible, leaving you with two smaller programs and some kind of shared functionality.

Kata Questions

- To what extent did the design decisions you made when writing the original programs make it easier or harder to factor out common code?

- Was the way you wrote the second program influenced by writing the first?
- Is factoring out as much common code as possible always a good thing? Did the readability of the programs suffer because of this requirement? How about the maintainability?

Posted by Dave Thomas (@PragDave) Dec 26th, 2013

Tweet 2 +1 0

« Kata05: Bloom Filters Kata03: How Big? How Fast? »

Comments

1 Comment pragdave Login

Sort by Best Share Favorite ★

Join the discussion…

Fred_Scuttle · 2 months ago

In answer to your questions:
- The design decisions I made for the first program, ignoring header lines, a method for removing extraneous characters from a numeric fields, made the second much easier.
- I simply modified the first program to address the second problem. Most of the same code was applicable.
- I'm a big believer in continuous refactoring. Number of lines to skip as well as field numbers were extracted as Java static final constants. I left other methods in place, such as the string trimming. That way, if the file format changes then the code modifications should be limited to changing the constants. If you're going to be using (essentially) the same code for different purposes then I believe that you should break out the common code into separate methods or even classes. I often create utility classes for specific functions, whether it's document conversion or cryptographic routines. I can readily move these classes to different projects or easily and conveniently reuse the methods as the application grows.
But it's sometimes hard to break junior programmers of the "cut and paste" habit. I always tell them that if they're cutting and pasting then they're doing something wrong. Invest the time to break out the common code into a separate method and ALWAYS document appropriately. We've actually got a target in our ant build.xml called javadoc. The idea is that, if everyone plays by the rules and documents code properly, then we have the potential for code re-use. And if it means that sometimes you have to use over-loaded methods then there's absolutely nothing wrong with that in my book.

· Reply · Share ›

ALSO ON PRAGDAVE WHAT'S THIS?

Kata05: Bloom Filters

3 comments • 3 months ago

Manh — But In the paper ,It different from conventional bloom filtersYou've read the paper "Multi-dimensional Range Query for

The 'Language' in Domain-Specific Language Doesn't Mean English (or

2 comments • 3 months ago

pragdave — I think I disagree, simply because business folk really don't want to read and audit these tests, whatever

Telling, Asking, and the Power of Jargon

7 comments • 3 months ago

pragdave — Love the "civilian casualty" phrase.I agree that it isn't binary, and my

Kata06: Anagrams - CodeKata

1 comment • 3 months ago

MarkF — To offer another point of comparison, this can be done with just 4

Recent Posts