KATALYST BROWSE ABOUT

Leap Year

Difficulty: Beginner

Estimated Duration: 1 hour

Introduction

This is another classic, usually attempted as a follow-on from FizzBuzz and normally considered a slight step up in difficulty because of the slightly more complicated rules. In practice, most developers can hold an entire solution to FizzBuzz in their working memory, but the Leap Year algorithm is a greater cognitive load, so it's normally not possible in one go.

This starts to reveal the power of baby-steps TDD. An algorithm that by itself feels like a bit of a challenge can be split into small, easy steps. As a bonus, since you're test-driving, you'll have living documentation and a full suite of regression tests once you've finished!

Carefully choosing the next test is essential - if you find yourself doing too large a step at any point, ask yourself: have you picked the right test case? You might need to wind back more than one step before you find an easier route. Like with other katas, success on this kata doesn't just mean that "it works": you can do it again and again to refine your approach (see Defining "Done" below).

Instructions

Implement a method that checks if a year is a leap year.

All the following rules must be satisfied:

- A year is not a leap year if not divisible by 4
- A year is a leap year if divisible by 4
- A year is a leap year if divisible by 400
- A year is not a leap year if divisible by 100 but not by 400

Examples:

- 1997 is not a leap year (not divisible by 4)
- 1996 is a leap year (divisible by 4)
- 1600 is a leap year (divisible by 400)
- 1800 is not a leap year (divisible by 4, divisible by 100, NOT divisible by 400)

The method should return true if a year is a leap year, and false if it is not.

Credit: Coding Dojo

Defining "Done"

How do you know when a kata is finished?

Firstly, it's not a one-shot thing. Some developers practice the same kata hundreds of times, each time trying a slightly new route, or a different technique.

Something that is important but sometimes overlooked is internalising the process of TDD. Try to always follow the principles while implementing the kata, even if it feels unnatural at first. If you get stuck and find you need to break one of the principles to get going again, that's fine, but make a note; it's an area for improvement. The next time you do the kata with more experience, you might find that it has taken care of itself.

Solutions

- C# by Tomaz Tekavec
- F# by Tomaz Tekavec
- Ruby by Tomaz Tekavec

Videos

Sandro Mancuso

Codurance Screencasts: The Leap Year Kata

Tomaz Tekavec

The Leap Year Kata with keyboard only

Tomaz Tekavec

The Leap Year Kata with mouse and keyboard

Classicist TDD

Algorithm Design

Psst!

Katalyst is in beta and we'd love your feedback.

Email us at katalyst@codurance.com and let us know what you think.

We are hiring

Are you looking for autonomy, mastery and purpose in your career? We are looking for people that share the same values of pragmatism, professionalism and transparency that we do.

Learn more

Software is our passion.

We are software craftspeople. We build well-crafted software for our clients, we help developers to get better at their craft through training, coaching and mentoring, and we help companies get better at delivering software.

Company Registration No: 8712584