Java With No Design Patterns

By Roger Keays, 13 September 2015

What have I done?

I look down at the code I have just written. It's just **one file** with **one static function** and **four unit tests**. There are no factories, no singletons, no dependency injection, no local or remote interface, no containers, no special annotations, no xml configuration, no inheritence hierarchy, no fancy polymorphism, no abstraction, no black magic, no UML diagrams, no dependencies, no interfaces, no mocks and no side effects.

It's just **code and tests**.

```
public class ProcessBeacon implements APILayer {

    public static boolean processBeacon(Long id, String email, String clier
       // code goes here
    }

    @Test
    public void processing_a_beacon_logs_an_action_in_the_database() {
      // test goes here
    }

    @Test
    public void processing_a_beacon_increments_the_open_count() {
      // test goes here
    }

    @Test
    public void processing_a_beacon_validates_the_recipients_email_address(
      // test goes here
    }

    @Test
```

```
        public void processing_a_beacon_resets_the_recipients_bounce_count() {
          // test goes here
        }
    }
```

What has gotten into me? How dare I believe I can just write code like that? Without consideration for the greater scheme of object oriented nirvana. Without spending hours philosophising about which class in the universal hierarchy of things my function belongs to. Without considering how to perfectly name it, and how to access and instantiate it.

Shame on you. A **beginner** could have done what you just did.

Yes, a beginner could have done it and they would have been way ahead of us developers with over a decade of experience who have lost themselves in the storm of object-oriented design patterns.

I've always had this uncomfortable feeling that something was wrong with the way I code. A feeling that *I don't know what I am creating*. I visualise my applications as complex machine components interacting with each other in fascinating ways. I marvel at the intricacy of how they must be perfectly polished and oiled in order to mesh together. I picture the beauty of how the software agents seamlessly handle the docking of polymorphic components.

It's a fantastic image, but it fails to answer my nagging inquiry... **what IS it**?

Is it a machine? Is it a pattern? Is it a virtual life form? Or is it just a jumble of machine code?

In a way, my switch to a more basic, functional style of programming has answered my question. Object oriented systems do attempt to create a sort of virtual life form, but this is incongruent with the way computers actually work (and one of the reasons object-oriented design fails so badly at concurrency and multithreading).

Computers are Turing Machines which work by streaming long lists of operations which transform data held in the CPU registers. In other words, they simply perform ***operations*** on ***data*** and send the results to the user.

Now think about how functional programming works. With functional programming you basically chain operations which transform data provided as the input to the function. In other words, you simply perform ***operations*** on ***data*** and send the results to the user.

Are you starting to see a pattern?

Business applications are hugely dominated by operations on data, but unfortunately we've been so sucked into modeling we forget to write code that actually **DOES STUFF**. Instead we spend most of our time figuring out how to instantiate objects and arguing about it on the Internet.

So I've broken all the rules of object oriented design and am implementing all my operations as **stateless static methods**, one per file with accompanying test cases. I no longer waste my time

modeling, I don't have to think about how to access the function (just CALL it!), I don't have to deliberate about where to put new functions and I can spend my somewhat limited brainpower on writing code that actually does something useful.

After all... models look great *but they don't DO much*.

About Roger Keays

Roger Keays is an artist, an engineer, and a student of life. He has no fixed addressand has leftfootprints on 40-something different countries around the world. Roger is addicted to surfing. His other interests are music, psychology, languages, the proper use of semicolons, and finding good food.