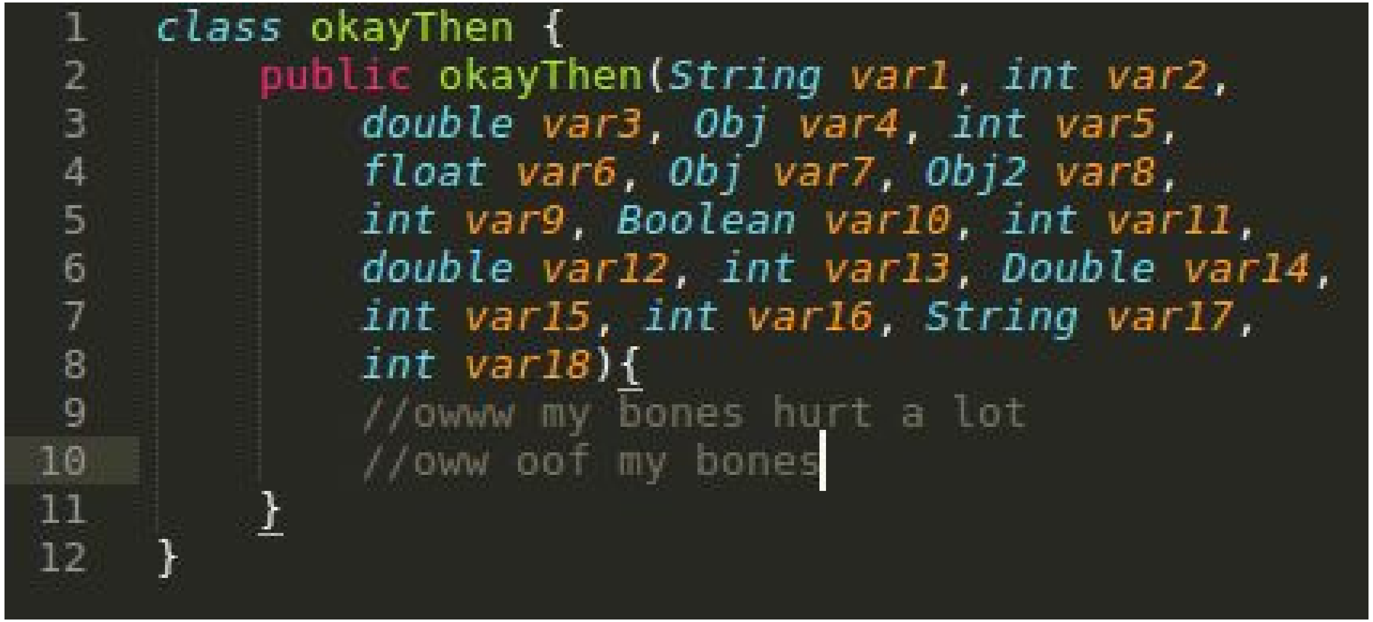
Builder Design Pattern

By Robin Shafto and Christian Fusco

Has this ever happened to you?



```
class okayThen {
     public okayThen(String var1, int var2,
          double var3, Obj var4, int var5,
          float var6, Obj var7, Obj2 var8,
          int var9, Boolean var10, int var11,
          double var12, int var13, Double var14,
          int var15, int var16, String var17,
          int var18){
          //owww my bones hurt a lot
          //oww oof my bones
     }
}
```

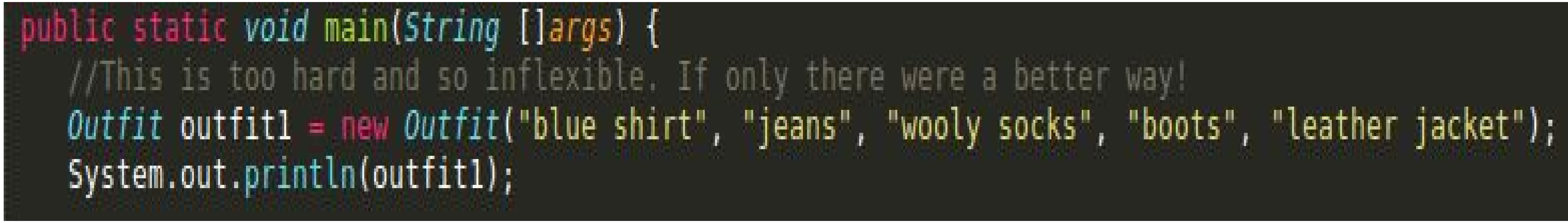
I’m so sorry if it has.

- The Design Problem
  - Too many variables to set
  - Ugly line breaks
  - Ordering matters
  - Imagine needing five of those constructors
  - Exposed getters and setters

Using a Builder to pretend that constructor never happened

- A class that hides another class's nasty parts
- Use a default constructor for initial values
- Use setters to change parts of it
- Call Builder.build() and get your object
    - Commonly used builds get their own sub-builders

I CAN'T EVEN MAKE IT LOOK NICE ON THE SLIDE



```
public static void main(String []args) {
    //This is too hard and so inflexible. If only there were a better way!
    Outfit outfit1 = new Outfit("blue shirt", "jeans", "wooly socks", "boots", "leather jacket");
    System.out.println(outfit1);
```

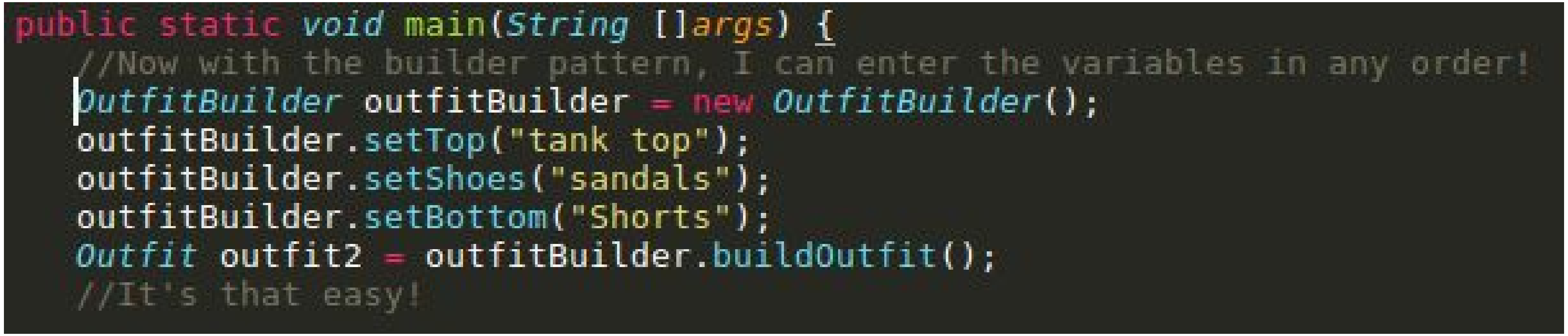
Fires start, sirens go off in the distance, every developer on a laptop is crying out in agony

There is no peace.

There is no happiness.

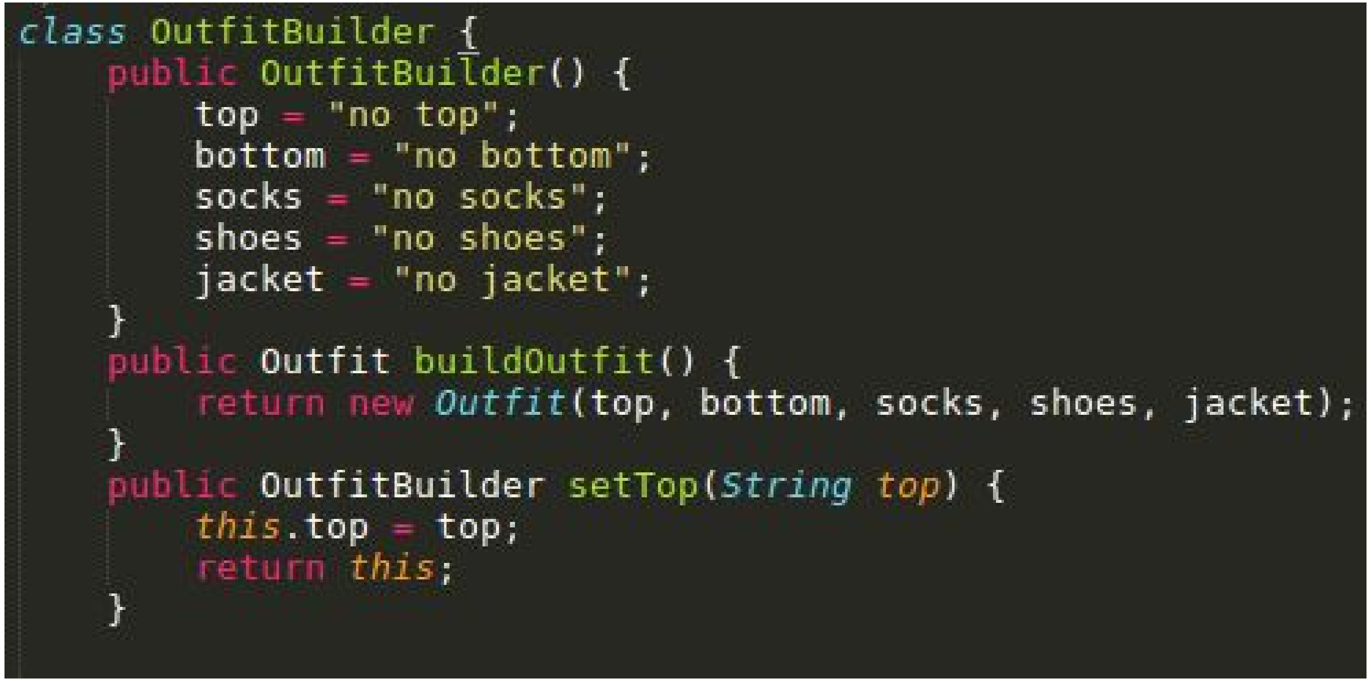
Only fear.

That’s like a breath of fresh air



```
public static void main(String []args) {
    //Now with the builder pattern, I can enter the variables in any order!
    OutfitBuilder outfitBuilder = new OutfitBuilder();
    outfitBuilder.setTop("tank top");
    outfitBuilder.setShoes("sandals");
    outfitBuilder.setBottom("Shorts");
    Outfit outfit2 = outfitBuilder.buildOutfit();
    //It's that easy!
```

Only go through great pain once



```java
class OutfitBuilder {
    public OutfitBuilder() {
        top = "no top";
        bottom = "no bottom";
        socks = "no socks";
        shoes = "no shoes";
        jacket = "no jacket";
    }
    public Outfit buildOutfit() {
        return new Outfit(top, bottom, socks, shoes, jacket);
    }
    public OutfitBuilder setTop(String top) {
        this.top = top;
        return this;
    }
```

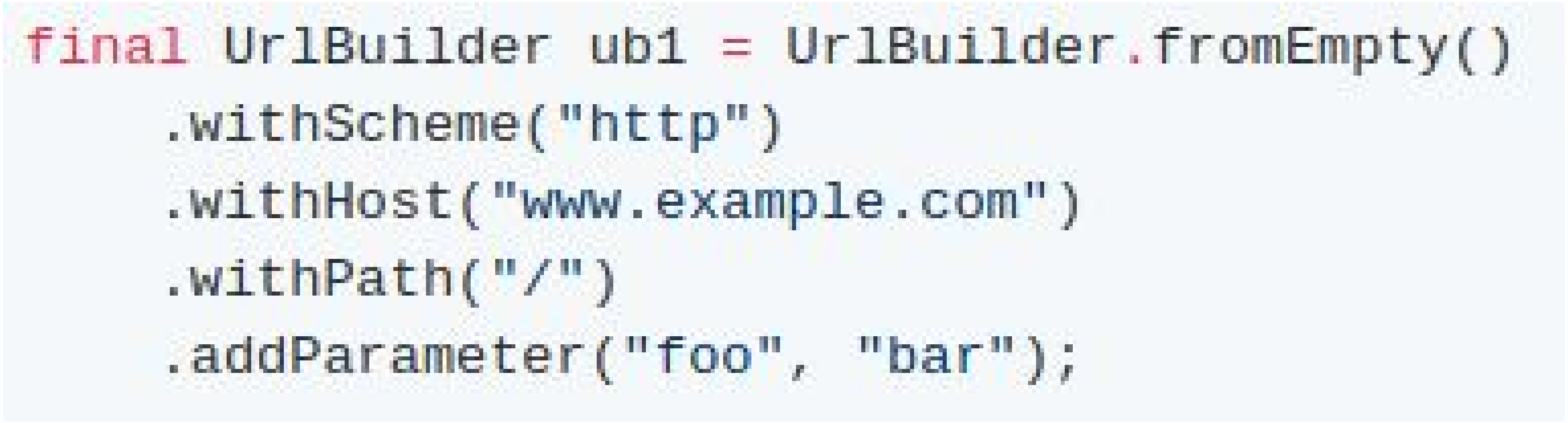
But will you ever use this?

- Heck yeah you will.
- You’ve probably been using it already.

XML Builders in JavaScript

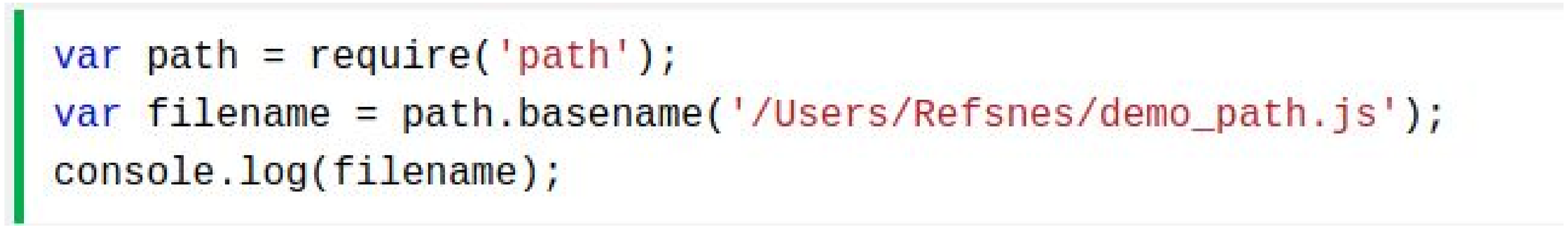
```javascript
var builder = require('xmlbuilder');
var xml = builder.create('root')
  .ele('xmlbuilder')
    .ele('repo', {'type': 'git'}, 'git://github.com/oozcitak/xmlbuilder-js.git')
  .end({ pretty: true});

console.log(xml);
```

URL Builders in Java



```java
final UrlBuilder ub1 = UrlBuilder.fromEmpty()
    .withScheme("http")
    .withHost("www.example.com")
    .withPath("/")
    .addParameter("foo", "bar");
```

Built in Path builder for NodeJS!!!!



```
var path = require('path');
var filename = path.basename('/Users/Refsnes/demo_path.js');
console.log(filename);
```

Recapping the benefits

- Builder classes will set variables by default
- Flexible, consistent objects
- Setters are easier to read than a constructor
- The order that the variables are set does not matter

Tradeoffs

- Only useful when there are many variables
- You have to work with more classes
- If too many presets are needed, a different class for each could be overkill

Thank u and goodnight.

https://github.com/oozcitak/xmlbuilder-js

https://github.com/mikaelhg/urlbuilder

https://github.com/jinder/path