Day 16: The prototype design pattern

The prototype pattern just involves generating new objects from existing ones via `copy` calls. In Scala, you get a powerful `copy` method for free on any case class!

```scala
case class ABC(a: Int, b: Int, c: Int)
val a1 = ABC(1, 2, 3)
val a2 = a1.copy(a = 4, b = 5)

// true
a1.isInstanceOf[ABC] && a2.isInstanceOf[ABC]

// false
a1 == a2

// nested is worse to deal with
case class DEF(d: Int, e: Int, f: ABC)
val d1 = DEF(1, 2, ABC(3, 4, 5))
d1.copy(f = d1.f.copy(b = 2))

```

This is great, except for the nested case, where you can get some pretty chunky boilerplate depending on how deeply nested your class is. For those cases, consider the `lense pattern`, which will be covered later in this series.