Updated: 2021-03-25

4.13
Lenses

Northeastern
University

Inverted *map*

- We all know that for a functor *F[A]*, the map method will be defined thus:
    - *def map[B](f: A=>B): F[B]*
- But, what if we were to have a method *x*:
    - *def x[B](f: B=>A): F[B]*
- What kind of a weird method is this *x*? Should we call it *unmap*? And how might we describe that function *f*? It's kind of backwards, right?

What is a lens?

- A lens is essentially a functional way of accessing the fields of an object.
  - Normally, it's a mutating method but I don't think it has to be.
- Suppose that you have an *Employee* class:
  - `case Class Employee(name: String, salary: Int)`
  - You want the salary field to be private:
  - `case Class Employee(name: String, private val salary: Int)`
  - So, something like `employee.salary` won't compile because the *salary* field is private.
  - Nevertheless, some people like the employee's manager need to be able to see this field. So, we could generate a function of type `Employee => Int` that will extract the *salary* information from the given *Employee* record. You would only be able to generate this function if you had the appropriate credentials. Alternatively, a function `Int => Employee => Employee` would allow you to create a copy of an employee with a different salary (see next slide).
  - We would call this kind of function a *lens* function.

What is a lens (2)?

- Perhaps a more typical use of a lens function would be to copy an *Employee* but give the new copy a different *salary*.

```
scala> val employee = Employee("Robin", 1000000)
employee: Employee = Employee(Robin,1000000)

scala> val updateSalary: Int=>Employee=>Employee = salary => employee =>
Employee(employee.name, salary)
updateSalary: Int => (Employee => Employee) =
$$Lambda$953/1599488589@237b2852

scala> updateSalary(1100000)(employee)
res1: Employee = Employee(Robin,1100000)
```

Comparer

- I have an open-source project called ***Comparer***:
  - if you did *lab-sorted* with me, you'll be familiar with the idea).
  - There is a method called ***snap***:

    ```
    def snap[U](f: U => T): Comparer[U] = u1 => u2 => self(f(u1))(f(u2))
    ```

  - It's like the ***unmap*** method we saw before.
  - The *U=>T* function *f* that it takes as a parameter is essentially a lens function: given a *U*, it will extract a *T*.

```
val ic = implicitly[Comparer[Int]]
val comparerY: Comparer[DateJ] = ic.snap(_.year)
val comparerM: Comparer[DateJ] = ic.snap(_.month)
val comparerD: Comparer[DateJ] = ic.snap(_.day)
val comparer: Comparer[DateJ] = comparerY orElse comparerM orElse comparerD
```