DDG University

Informal Goals

1. Learn new things related to technology.
2. Learn from each other.
3. Foster inter-team building.
4. To become better engineers.

Search for DDG University in Asana.

Structure and Interpretation of Computer Programs *(SICP)*

by Harold Abelson and Gerald Jay Sussman

2.1 Introduction to Data Abstraction

1. ***Data Abstraction***

A Little More Lisp

- *cons*
- *car*
- *cdr*

cons, car, cdr

```
(define pair (cons 100 300))

(car pair)
> 100

(cdr pair)
> 300
```

Wishful Thinking

Lets suppose we have:

- `(make-rat ⟨n⟩ ⟨d⟩)` returns the rational number whose numerator is the integer ⟨n⟩ and whose denominator is the integer ⟨d⟩.
- `(numer ⟨x⟩)` returns the numerator of the rational number ⟨x⟩.
- `(denom ⟨x⟩)` returns the denominator of the rational number ⟨x⟩.

Sounds like a contract or...

an ***interface!***

Our Rational Number Code

```scheme
(define (add-rat x y)
  (make-rat
    (+ (* (numer x) (denom y))
       (* (numer y) (denom x)))
    (* (denom x) (denom y))))

(define (sub-rat x y)
  (make-rat
    (- (* (numer x) (denom y))
       (* (numer y) (denom x)))
    (* (denom x) (denom y))))

(define (mul-rat x y)
  (make-rat
    (* (numer x) (numer y))
    (* (denom x) (denom y))))

(define (div-rat x y)
  (make-rat
    (* (numer x) (denom y))
    (* (denom x) (numer y))))
```

Rational Number Implementation

```scheme
(define (make-rat n d) (cons n d))
(define (numer x) (car x))
(define (denom x) (cdr x))
```

Looks a little like a... class!

Abstraction Barriers

```
----------[Programs that use rational numbers]-----------

           Rational numbers in problem domain

-------------------[add-rat sub-rat ...]-----------------

      Rational numbers as numerators and denominators

------------------[make-rat numer denom]------------------

                 Rational numbers as pairs

-----------------------[cons car cdr]--------------------

               However pairs are implemented
```

An Alternate Implementation of Our "Class"

```
(define (cons x y)
  (define (dispatch m)
    (cond ((= m 0) x)
          ((= m 1) y)
          (else
           (error "Argument not 0 or 1: CONS" m))))
  dispatch)

(define (car z) (z 0))
(define (cdr z) (z 1))
```

No tradition data structures here. Everything is stored in deferred procedures.

Wrapping-up

- Data abstraction...

That's all for section 2.1.

Thanks!