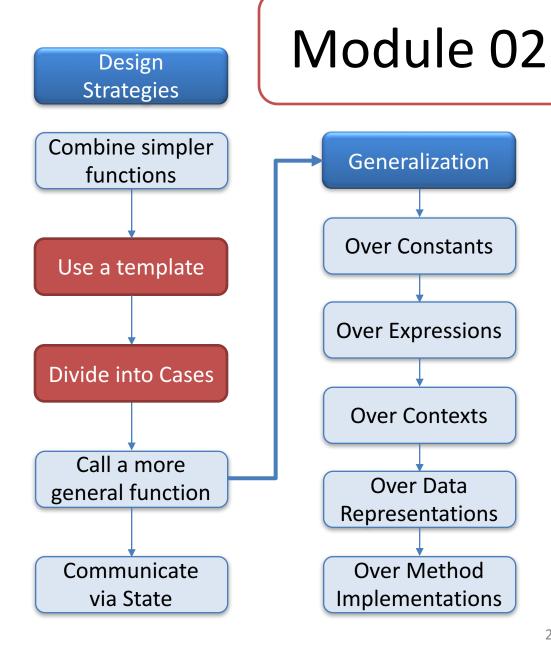
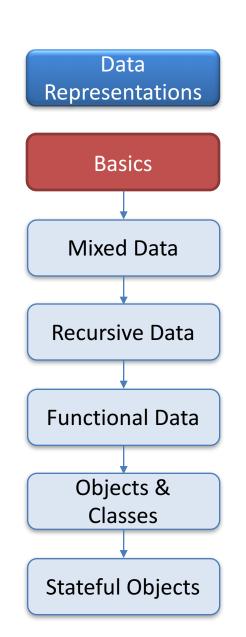
Design Strategies 2: Using a template

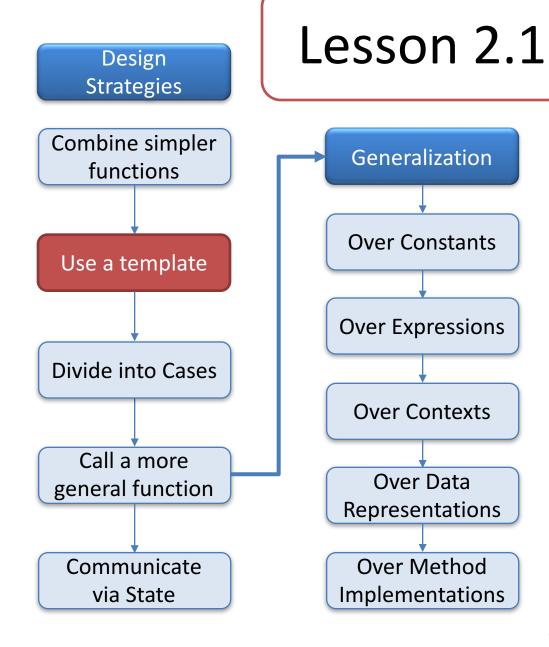
CS 5010 Program Design Paradigms
Lesson 2.1



Data Representations Basics Mixed Data **Recursive Data Functional Data** Objects & Classes Stateful Objects







Introduction

- In this lesson, we will show how to take apart non-scalar data using the observer template for that kind of data.
- This is the strategy you will use for the vast majority of your functions.

Let's see where we are

The Six Principles of this course

- 1. Programming is a People Discipline
- 2. Represent Information as Data; Interpret Data as Information
- 3. Programs should consist of functions and methods that consume and produce values
- 4. Design Functions Systematically
- 5. Design Systems Iteratively
- 6. Pass values when you can, share state only when you must.

The Function Design Recipe

- 1. Data Design
- 2. Contract and Purpose Statement
- 3. Examples and Tests
- 4. Design Strategy
- 5. Function Definition
- 6. Program Review

Design Strategies

- 1. Combine simpler functions
- Use template for <data def> on <vble>
- 3. Divide into cases on <condition>
- 4. Use HOF <mapfn> on <vble>
- 5. Call a more general function

Use a destructor template

- Used when the problem can be solved by examining a piece of non-scalar data.
- Slogan:

The shape of the data determines the shape of the program.

What does it mean to "examine" a piece of data?

- If the data is compound data, this means extracting its fields.
- If the data is itemization data, this means determining which variant the data is.
- If the data is mixed data, this means determining which variant the data is, and then extracting its fields, if any.
- Every data definition includes a template that shows how this examination process is to be organized.
- Writing a function using a template is accomplished by filling in the blanks in the template.
 - Definition of "filling in the blank" to come in Slide 11.

From Template to Function Definition

Recipe for Using a Template

- 1. Make a copy of the template and uncomment it
- 2. Fill in the function name and add more arguments if needed
- 3. The strategy is "Use template for <data def> on <vble>," where <data def> is the kind of data you are taking apart, and <vble> is the variable whose value you are looking at.
- 4. Fill in the blanks in the template by combining the arguments and the values of the fields using simpler functions.

Example: book-receipts

```
;; book-receipts : Book NonNegInt -> NonNegInt
;; GIVEN: a Book and the number of copies sold
;; RETURNS: the total receipts from the sales of the
;; given book. Ignores the number of copies on hand.
;; EXAMPLE:
;; (book-receipts
;; (make-book "Felleisen" "HtdP2" 13 2795) 100)
;; = 279500
```

To do this, we'll need to look inside the Book to see its price, so we'll use the Book template

1. Make a copy of the template and uncomment it

```
(define (book-fn b)
   (...
    (book-author b)
     (book-title b)
     (book-on-hand b)
     (book-price b)))
```

2. Fill in the function name and add more arguments if needed

```
(define (book-receipts b sales)
  (...
   (book-author b)
   (book-title b)
   (book-on-hand b)
   (book-price b)))
```

3. Write down the strategy

```
;; STRATEGY: Use template for Book on b.
(define (book-receipts b sales)
   (...
        (book-author b)
        (book-title b)
        (book-on-hand b)
        (book-price b)))
```

4. Fill in the blanks in the template

```
;; STRATEGY: Use template for Book on b.
(define (book-receipts b sales)
   (* (book-price b) sales))

Things we didn't use:
   (book-author b)
   (book-title b)

We said:
"4. Fill in the blanks in the template by combining the arguments and the values of the fields using simpler functions."
```

(book-on-hand b)

That's OK!

Example: next state of traffic light

```
;; DATA DEFINITION:
;; a TrafficLightState (TLState) is one of
;; -- "red"
;; -- "yellow"
;; -- "green"
;; INTERPRETATION: self-evident
```

Contract and Purpose Statement

```
;; next-state : TLState -> TLState
;; GIVEN: a TLState
;; RETURNS: the TLState that follows the given TLState
;; EXAMPLES:
;; (next-state "red") = "green"
;; (next-state "yellow") = "red"
;; (next-state "green") = "yellow"
```

1. Make a copy of the template and uncomment it

```
(define (tls-fn state)
  (cond
    [(string=? state "red") ...]
    [(string=? state "yellow") ...]
    [(string=? state "green") ...]))
```

2. Fill in the function name and add more arguments if needed

```
(define (next-state state)
  (cond
    [(string=? state "red") ...]
    [(string=? state "yellow") ...]
    [(string=? state "green") ...]))
```

3. Fill in the strategy

```
;; STRATEGY: Use template for TLState on state

(define (next-state state)
  (cond
    [(string=? state "red") ...]
    [(string=? state "yellow") ...]
    [(string=? state "green") ...]))
```

```
;; STRATEGY: Use template for TLState on state

(define (next-state state)
  (cond
    [(string=? state "red") ...]
    [(string=? state "yellow") ...]
    [(string=? state "green") ...]))
```

What is the answer for "red"?

;; STRATEGY: Use template for TLState on state

What is the answer for "red"?

Answer (from examples): "green"

;; STRATEGY: Use template for TLState on state

What is the answer for "yellow"?

Answer (from examples): "red"

```
;; STRATEGY: Use template for TLState on state

(define (next-state state)
  (cond
    [(string=? state "red") "green"]
    [(string=? state "yellow") "red"]
    [(string=? state "green") "yellow"]))

Answer (from examples): "yellow"]
```

Working with other kinds of data

- We've seen how to use templates for compound data and itemization data
- Mixed data works the same way.
- Copy the template, uncomment it, and fill in the missing pieces. That's it!
- If you've thought hard enough about your function, filling in the blanks is easy.

What can you put in the blanks?

- We said: Fill in the blanks in the template by combining the arguments and the values of the fields using simpler functions.
- This means :
 - You don't have to use all of the fields
 - You can use a field twice
 - You don't have to use the fields "in order"
- But it has to be simple, as in Lesson 1.7

Next Steps

- Study 02-1-book-receipts.rkt and 02-2-traffic-light.rkt in the Examples folder.
 - Be sure to finish the **previous-state** example in 02-2-traffic-light.rkt
- If you have questions or comments about this lesson, post them on the discussion board.
- Do the Guided Practices
- Go on to the next lesson.