UMass Boston Computer Science

CS450 **High Level Languages** (section 2)

Compound Data Definitions

Monday, September 25, 2023

Logistics

- ## HW 1 in
  - ~~due: Sun 9/24 11:59 pm EST~~
  - Files should not start `big-bang` loop automatically

- ## HW 2 out
  - due: Sun 10/1 11:59 pm EST

- ## STYLE notes
  - Use comments to explain code <u>if needed</u>, BUT …
    - … the best code needs no comments `(not (string? str))`

(not a great variable name)

  - Redundant comments makes code <u>harder to read</u>
    - More comments ≠ "better"

```
; checks if str is a string
((not (string? str))
```

  - Also, no commented-out code

Falling Ball Example

```
;; A WorldState is a Non-negative Integer
;; Interp: Represents the y Coordinate of the center of a
;;         ball in a `big-bang` animation.
```
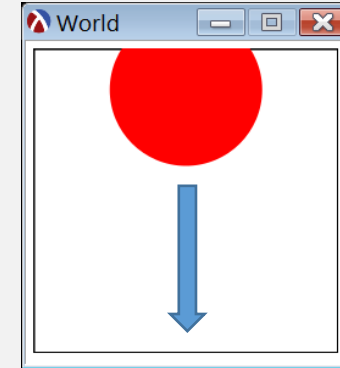
What if the **ball can also move side-to-side**?

WorldState would need <u>two</u> pieces of data:
the **x** *and* **y** coordinates

```
;; A WorldState is an Integer ...
;; ... and another Integer???
```

We need a way to create **compound data**
i.e., **a new data definition that <u>combines</u>**
values from other data defs

Kinds of Data Definitions

- Basic data
  - E.g., **numbers, strings,** etc
- Intervals
  - Data that is from a **range of values,** e.g., [0, 100)
- Enumerations
  - Data that is **one of a list of possible values**, e.g., "green", "red", "yellow"
- Itemizations
  - Data value that can be from a **list of possible other data definitions**
  - E.g., <u>either</u> a string or number (Generalizes enumerations)
- **Compound Data**
  - Data that is a **combination of values from other data definitions**

Falling Ball Example

```
;; A WorldState is a
(struct world [x y])
;; where
;; x: Integer - represents x coordinate of ball in animation
;; y: Integer - represents y coordinate of ball
```

a **struct** definition creates a new kind of **compound data**

**Instances** of the **struct** are values of that kind of data

```
(define INITIAL-STATE (world 0 0))
```

Parts of a `struct` definition

field-names

name

```
(struct world [x y])
```

Same as name

field-names
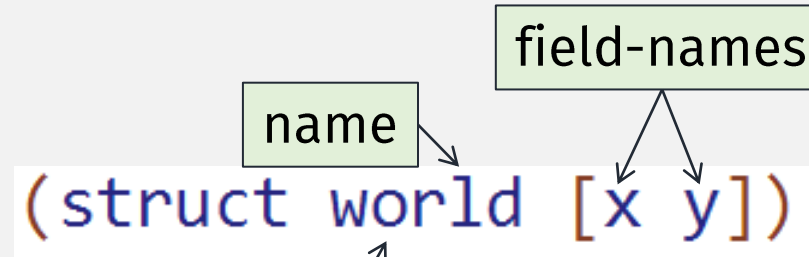
(Implicitly) defines:

- A **constructor** function → `world`
  - Creates instances of the struct
- **Accessor** functions → `world-x, world-y`
  - Get an instance's field value
- A **predicate** → `world?`
  - Returns true for struct instances
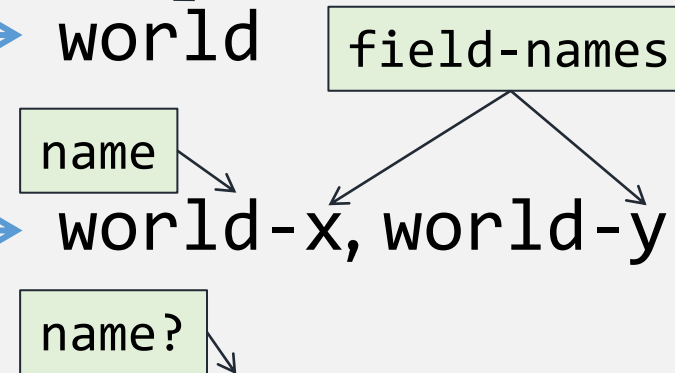
name

name?

Function Design Recipe

1. **Name**

2. **Signature** – <u>types</u> of the **function input(s)** and **output**

3. **Description** – <u>explain</u> (in **English prose**) the **function behavior**

4. **Examples** – <u>show</u> (using `rackunit`) the **function behavior**

5. **Code** – <u>implement</u> the **rest of the function** (arithmetic)

6. **Tests** – <u>check</u> (using `rackunit`) the **function behavior**

Function Design Recipe

1. **Name**

2. **Signature** – <u>types</u> of the **function input(s)** and **output**

3. **Description** – <u>explain</u> (in **English prose**) the **function behavior**

4. **Examples** – <u>show</u> (using `rackunit`) the **function behavior**

5. **Template** – <u>sketch out</u> the **function structure** (using input's **Data Definition**)

6. **Code** – <u>implement</u> the **rest of the function** (arithmetic)

7. **Tests** – <u>check</u> (using `rackunit`) the **function behavior**

8

Template for Compound data

- A **function that consumes compound data** must
    - <u>extract</u> **the individual pieces,** using accessors
    - <u>combine</u> **them,** with arithmetic

```
;; A WorldState is a
(struct world [x y])
;; where
;; x: Integer - represents x coordinate of ball in animation
;; y: Integer - represents y coordinate of ball
```

```
;; TEMPLATE for world-fn: WorldState -> ???
(define (world-fn w)
    .... (world-x w) ....
    .... (world-y w) ....)
```

Code demo

- Moving ball
  - Both x and y coordinate can change
  - With mouse movement
  - (and keyboard directions?)

Check-In Quiz 9/25
on gradescope

(due 1 minute before midnight)