第5回
種々のデータと構造体

情報処理演習II

Literacy

様々のデータ

- 単純データ
  - 数値　number　・・・ 数値情報
  - ブール値　boolean　・・・ true/false 値
  - シンボル　symbol　・・・ 記号情報
  - 文字列　string　・・・ 文字列情報
- 合成データ:
  - 構造体　struct
    1. ユーザが自ら定義した構造体
    2. DrScheme に組み込み済みの構造体: posn など
  - リスト　list

Literacy

例題

Literacy

例題1. シンボル

- x の値から，3種類のシンボル ('Cold, 'Warm, 'Hot) のどれかを出力する関数 judge を定義し，実行する

x ≦ 20　→ 'Cold
20 < x ≦ 30 → 'Warm
30 < x　→ 'Hot

シンボル

Literacy

judge 関数の入力と出力

x の値:
15

入力は
1つの数値

出力は
1つのシンボル

Literacy

judge 関数

「関数である」ことを示すキーワード

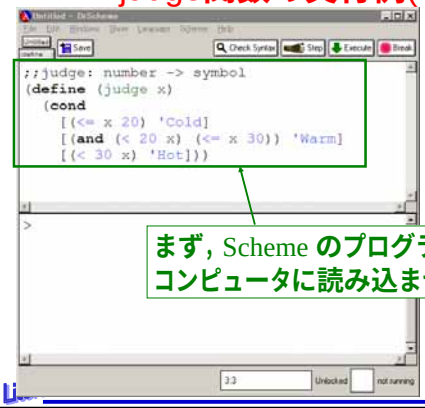
関数の名前

```
;; judge: number -> symbol
(define (judge x)
  (cond
    [(<= x 20) 'Cold]
    [(and (< 20 x) (<= x 30)) 'Warm]
    [(< 30 x) 'Hot]))
```

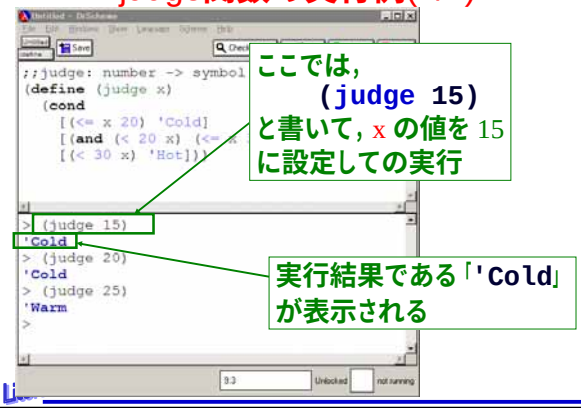
値を1つ受け取る (入力)

Literacy

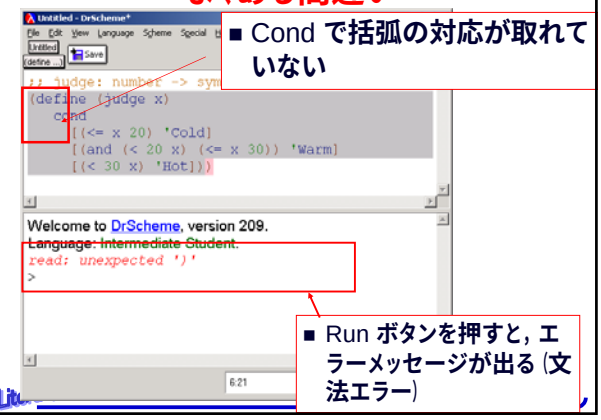
judge関数の実行例(1/2)



judge関数の実行例(2/2)



よくある間違い

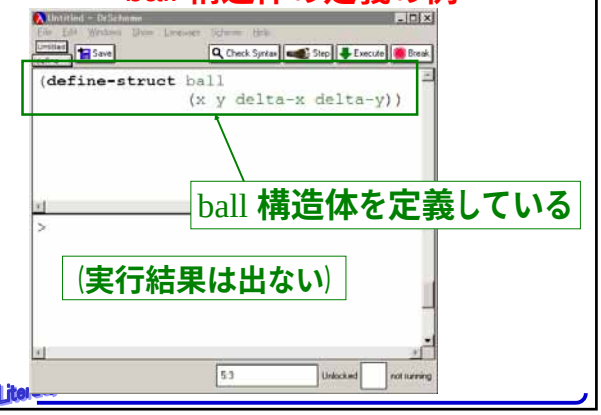


例題2．ユーザ定義の構造体

- ball 構造体(ball structure) を定義する
  - ball は，x, y, delta-x, delta-y の4つの属性から構成
  - 構造体の定義では「`define-struct`」を使用

x	位置
y	
delta-x	速度
delta-y	

ball 構造体の定義の例



ball 構造体

構造体の名前

```
(define-struct ball
        (x y delta-x delta-y))
```

属性の並び
（それぞれの属性にも名前がある）

ball 構造体

名前　　属性の並び

```
(define-struct ball (x y delta-x delta-y))
```

```
make-ball : (number number number number -> ball)
```
コンストラクタ

```
ball-x : (ball -> number)
ball-y : (ball -> number)
ball-delta-x : (ball -> number)
ball-delta-y : (ball -> number)
```
セレクタ

```
ball? : (anything -> boolean)
```
ball 構造体かを調べる

Literacy

例題3．ball の原点からの距離

- ボールの座標値から，原点からの距離を求める関数 `distance_to_0` を定義し，実行する
  - 例題2の ball 構造体を使用
  - 属性 x, y を取り出すために `ball-x`, `ball-y`（セレクタ）を使う

x
y
delta-x
delta-y

x, y：位置　　delta-x, delta-y：速度

Literacy

distance_to_0 関数の入力と出力

`a-ball` の値：
`(make-ball 3 4 0 0)`

入力 → `distance_to_0` → 出力 5

入力は ball 構造体　　出力は数値

Literacy

distance_to_0 関数の定義

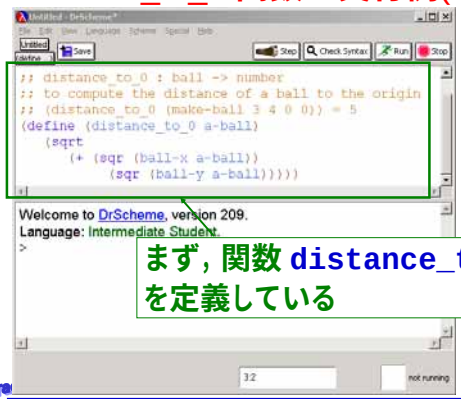
```
(define-struct ball (x y delta-x delta-y))
;; distance_to_0 : ball -> number
;; to compute the distance of a ball to
;; the origin
;; (distance_to_0 (make-ball 3 4 0 0)) = 5
(define (distance_to_0 a-ball)
   (sqrt
      (+ (sqr (ball-x a-ball))
         (sqr (ball-y a-ball)))))
```

「ball-x」は，x の値の取得のためのセレクタ

「ball-y」は，y の値の取得のためのセレクタ

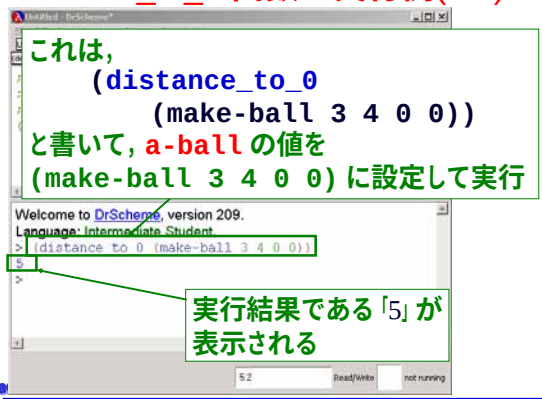
Literacy

distance_to_0 関数の実行例(1/3)



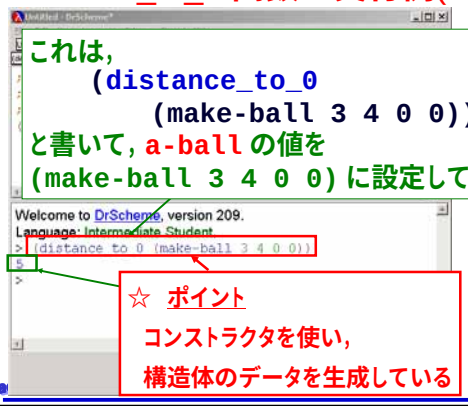
Literacy

distance_to_0 関数の実行例(2/3)



Literacy

distance_to_0 関数の実行例(3/3)



これは,
```
(distance_to_0
    (make-ball 3 4 0 0))
```
と書いて, a-ball の値を
(make-ball 3 4 0 0) に設定して実行

☆ ポイント
コンストラクタを使い,
構造体のデータを生成している

DrScheme に組み込み済みの構造体

■posn 構造体
–点データ: 属性 x, y

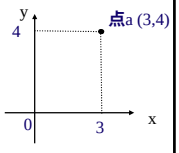
posn は基本的な組込みの構造体として最初から定義済なのであらためて定義しなくてよい

```
make-posn : (number number -> posn)
```
コンストラクタ

```
posn-x : (posn -> number)
posn-y : (posn -> number)
```
セレクタ

```
posn? : (anything -> boolean)
```
posn 構造体かを調べる

posn構造体



teachpack の draw.ss を使う

DrScheme では, グラフィックスの teachpack である draw.ss を提供

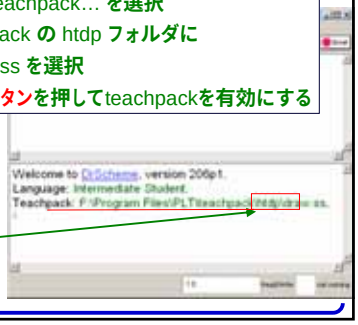
1. `draw-solid-line`: 始点と終点のposn構造体と色を受け取り線を描く。
2. `draw-solid-rect`: 左上角のposn構造体、幅、高さ、色を受け取り長方形を描く。
3. `draw-solid-disk`: 中心の posn 構造体、半径と色を受け取り円盤を描く。
4. `draw-circle`: 中心の posn 構造体、半径と色を受け取り円を描く。

それぞれ、描画に成功すれば trueを返す。
各描画演算には対応する clear-演算がある:
`clear-solid-line`、`clear-solid-rect`、
`clear-solid-disk`、`clear-circle`

詳しくは DrScheme の Help Desk にある draw.ss の説明を参照

teachpack の draw.ss を使う

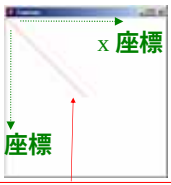
teachpack を読み込む (draw.ss の例)
Language → Add Teachpack… を選択
→ teachpack の htdp フォルダに
ある draw.ss を選択
→ Run ボタンを押してteachpackを有効にする



Teachpack の draw.ss が読み込まれたことを示す
(注:下線部はシステムごとに異なる)

描画演算のためのキャンバス

左上角が(0,0)




```
(draw-solid-line
    (make-posn 1 1) (make-posn 50 50)'red)
```
で描画された直線

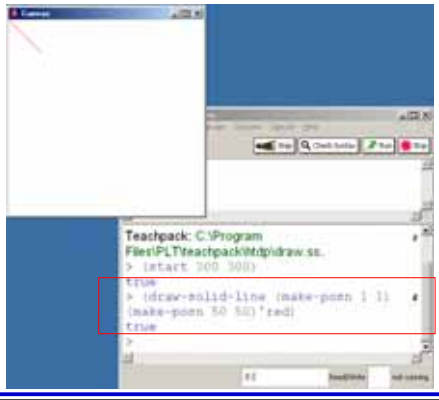
■ キャンバスでは, 左上角の座標が(0,0)
■ y座標の値は下にいくほど大きくなる

例題4. 簡単な絵を描く

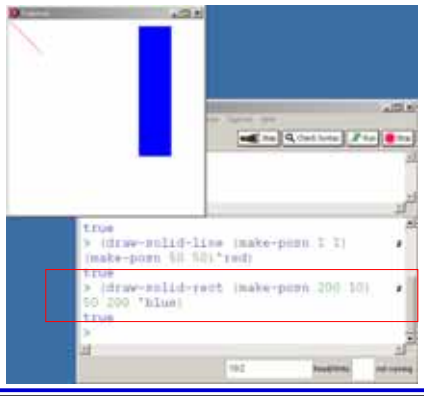
```
(start 300 300)    キャンバスを開く
(draw-solid-line (make-posn 1 1)
  (make-posn 50 50)'red)
      赤い線の描画
(draw-solid-rect (make-posn 200 10) 50 200
  'blue)
      青い長方形の描画（幅50で高さ200）
(draw-circle (make-posn 200 10) 50 'red)
      赤い円の描画（半径50）
(draw-solid-disk (make-posn 200 10) 50 'green)
      緑の円盤の描画（半径50で中心が長方形の頂点）
(draw-solid-string (make-posn 50 50) "Here")
      文字列(“Here”)の描画
(stop)    キャンバスを閉じる
```

実行例(1/6)

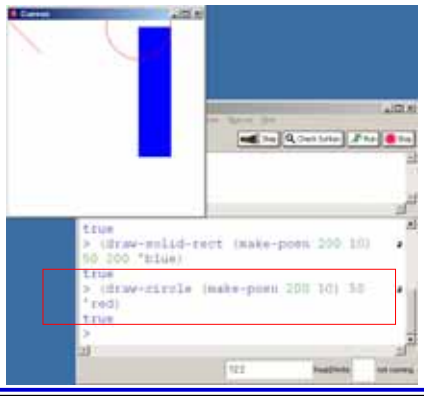
実行例(2/6)



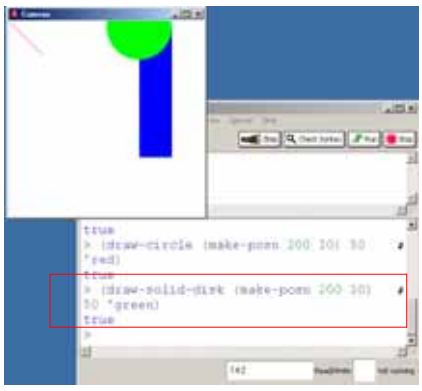
実行例(3/6)



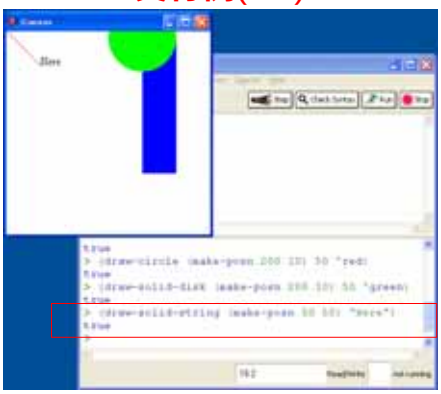
実行例(4/6)



実行例(5/6)



実行例(6/6)



Literacy

実習

Literacy

実習の進め方

- 資料を見ながら、「実習」を行ってみる
- その後、各自「課題」に挑戦する
  - 各自で自習 ＋ 巡回指導
  - 遠慮なく質問してください
- 自分のペースで先に進んで構いません

Literacy

DrScheme の使用

- DrScheme の起動
  プログラム → PLT Scheme → DrScheme
- 今日の実習では「Intermediate Student」に設定
  Language
  → Choose Language
  → Intermediate Student
  → OK ボタン

Literacy

DrScheme の使用 / ステップ実行

- プログラム実行の振る舞いを観察するツール
- 「定義用ウインドウ」のみを使用
  (通常の実行と異なる)
- 「Intermediate Student」に設定する必要あり
  Language → Choose Language
  → Intermediate Student
  → Run ボタン

Literacy

draw.ss teach pack のロード

- DrScheme の描画機能である draw.ss teachpack を使うために,

draw.ss teachpack をロードせよ

この操作は1回だけでよい
(次回からは自動的にロードされるようになる)

Literacy

実習1．数値かシンボルを出力

- x の値から，数値あるいはシンボルを出力する関数 **ast** を定義し，実行する
  - x ＞ 0 ならば： x の値を出力する
  - x ≦ 0 ならば： 「'*」を出力する

「実習1．数値かシンボルを出力」の手順

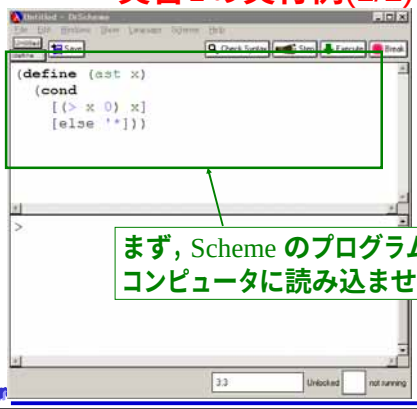
1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Run ボタンを押す

```
;; ast: number -> number or '*
(define (ast x)
  (cond
    [(> x 0) x]
    [else '*]))
```

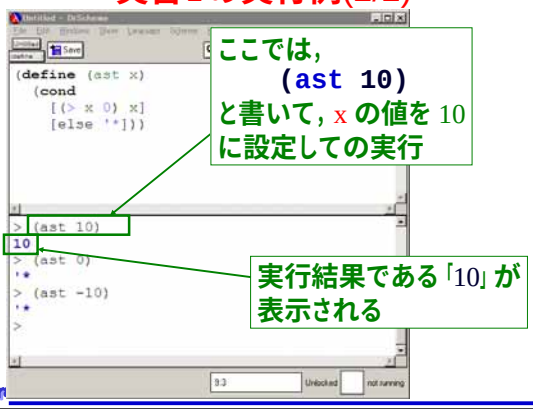
2. その後，次を「実行用ウインドウ」で実行しなさい

```
(ast 10)
(ast 0)
(ast -10)
```

実習1の実行例(1/2)



実習1の実行例(2/2)



ast 関数の入力と出力

x の値：
10

10

入力は
1つの数値

出力は
1つの数値
あるいはシンボル

ast 関数

「関数である」ことを
示すキーワード

関数の名前

```
;; ast: number -> number or '*
(define (ast x)
  (cond
    [(> x 0) x]
    [else '*]))
```

値を1つ受け取る（入力）

実習2．ステップ実行

- 関数 `distance_to_0` の実行過程
  - `(distance_to_0 (make-ball 3 4 0 0))` から 5 に至る過程を見る
  - DrScheme のステップ実行機能を使用する

```
(distance_to_0 (make-ball 3 4 0 0))
= (sqrt (+ (sqr (ball-x (make-ball 3 4 0 0)))
           (sqr (ball-y (make-ball 3 4 0 0)))))
= (sqrt (+ (sqr 3)
           (sqr (ball-y (make-ball 3 4 0 0)))))
= ...
= (sqrt (+ 9
           (sqr (ball-y (make-ball 3 4 0 0)))))
= (sqrt (+ 9
           (sqr 4)))
= ...
= (sqrt (+ 9 16))
= (sqrt 25)
= 5
```

Literacy

「実習2．ステップ実行」の手順

1. 次を「定義用ウインドウ」で，実行しなさい
   - Intermediate Student で実行すること
   - 入力した後に，Run ボタンを押す

例題3と同じ

```
(define-struct ball
    (x y delta-x delta-y))
;; distance_to_0: ball -> number
;; to compute the distance of a ball
;; to the origin
;; (distance_to_0 (make-ball 3 4 0 0)) = 5
(define (distance_to_0 a-ball)
   (sqrt
      (+ (sqr (ball-x a-ball))
         (sqr (ball-y a-ball)))))
(distance_to_0 (make-ball 3 4 0 0))
```

2. DrScheme を使って，ステップ実行の様子を確認しなさい （Step ボタン，Next ボタンを使用）
   - 理解しながら進むこと

ステップ実行したいので、定義用ウインドウだけを使う

Literacy

(distance_to_0 (make-ball 3 4 0 0)) から 5 に至る過程の概略

(distance_to_0 (make-ball 3 4 0 0)) 最初の式

```
= (sqrt (+ (sqr (ball-x (make-ball 3 4 0 0)))
           (sqr (ball-y (make-ball 3 4 0 0)))))
= (sqrt (+ (sqr 3)
           (sqr (ball-y (make-ball 3 4 0 0)))))
= ...
= (sqrt (+ 9
           (sqr (ball-y (make-ball 3 4 0 0)))))
= (sqrt (+ 9
           (sqr 4)))
= ...
= (sqrt (+ 9 16))
= (sqrt 25)
```

コンピュータ内部での計算

= 5 実行結果

(distance_to_0 (make-ball 3 4 0 0)) から 5 に至る過程の概略

```
(distance_to_0 (make-ball 3 4 0 0))
= (sqrt (+ (sqr (ball-x (make-ball 3 4 0 0)))
           (sqr (ball-y (make-ball 3 4 0 0)))))
= (sqrt (+ (sqr 3)
```

これは，

```
(define (distance_to_0 a-ball)
  (sqrt
    (+ (sqr (ball-x a-ball))
       (sqr (ball-y a-ball)))))
```

の a-ball を (make-ball 3 4 0 0) で置き換えたもの

```
= (sqrt (+ 9 16))
= (sqrt 25)
= 5
```

実習3．personal_info 構造体

`personal_info` 構造体の定義

構造体名　フィールド名の並び

```
(define-struct personal_info (name age address))
```

上記の定義によって以下が使えるようになる

- コンストラクタ
  - `(make-personal_info 値 値 値)`
- セレクタ
  - `(personal_info-name …)`
  - `(personal_info-age …)`
  - `(personal_info-address …)`
- `personal_info` かを調べる
  - `(personal_info? …)`

フィールド

personal_info
name
age
address

Literacy

「実習3．personal_info 構造体」の手順

1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Run ボタンを押す

```
(define-struct personal_info
                (name age address))
```

2. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Run ボタンを押す

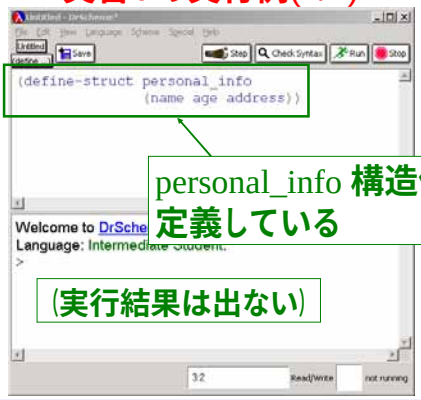
```
(define A (make-personal_info "Ken" 30 "kashii"))
(define B (make-personal_info "Mike" 25 "kashii"))
```

3. その後，次を「実行ウインドウ」で実行しなさい

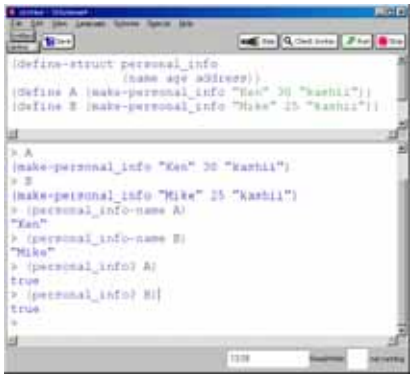
```
A
B
(personal_info-name A)
(personal_info-name B)
(personal_info? A)
(personal_info? B)
```

Literacy

実習3の実行例(1/2)

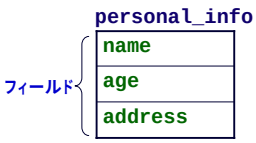


実習3の実行例(2/2)



実習4. 名簿データを調べる

- personal_info 構造体を入力とし，成人なら name を答え，未成年なら「“Anomymous”」を答える関数**check-name**を定義し，実行する
  - **personal_info 構造体データの age フィールドを調べ 20歳以上なら name フィールドデータを答え、20歳未満なら「"Anonymous"」と答える**



check-name 関数の入力と出力

```
(make-personal_info "Ken" 18 "Ropponmatsu")  "Anonymous"
(make-personal_info "Bill" 20 "Hakozaki")    "Bill"
```

入力は personal_info**構造体**

出力は 文字列 (string)

「実習4. 名簿データを調べる」の手順

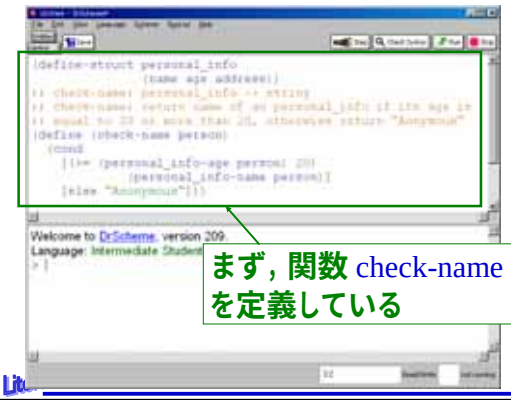
1. **次を「定義用ウインドウ」で，実行しなさい**
   - **入力した後に，Run ボタンを押す**

```
(define-struct personal_info
                (name age address))
;; check-name: personal_info -> string
;; check-name: return name of an personal_info
;; if its age is equal to 20 or more than 20,
;; otherwise return "Aonymous"
(define (check-name person)
  (cond
    [(>= (personal_info-age person) 20)
          (personal_info-name person)]
    [else "Anonymous"]))
```

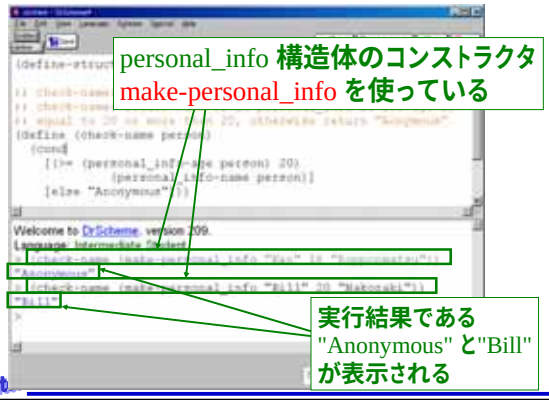
2. **その後，次を「実行用ウインドウ」で実行しなさい**

```
(check-name (make-personal_info "Ken" 18 "Ropponmatsu"))
(check-name (make-personal_info "Bill" 20 "Hakozaki"))
```

実行例



実行例



check-name プログラム

```
(define-struct personal_info
               (name age address))
;; check-name: personal_info -> string
;; check-name: return name of an personal_info
;; if its age is equal to 20 or more than 20,
;; otherwise return "Aonymous"
(define (check-name person)
  (cond
    [(>= (personal_info-age person) 20)
             (personal_info-name person)]
    [else "Anonymous"]))
```

personal_info 構造体のセレクタ

実習5．簡単な絵を描く

- DrScheme の描画機能 (draw.ss teachpack) を使って，簡単な絵を描く

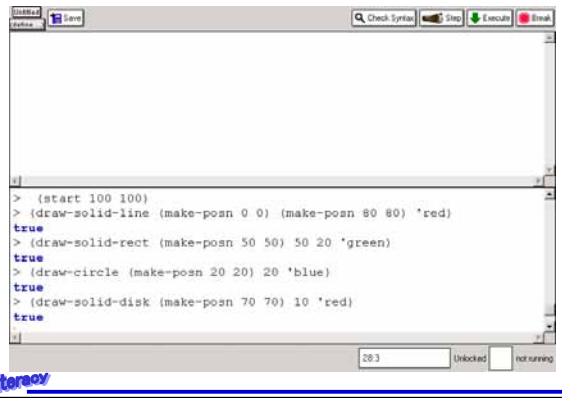
`start`:	「描画用ウインドウ」を開く
`draw-solid-line`:	線
`draw-solid-rect`:	四角形
`draw-solid-disk`:	塗りつぶされた円
`clear-solid-disk`:	一度描いた「塗りつぶされた円」を消す
`draw-circle`:	円
`stop`:	「描画用ウインドウ」を閉じる

「実習5．簡単な絵を描く」の手順

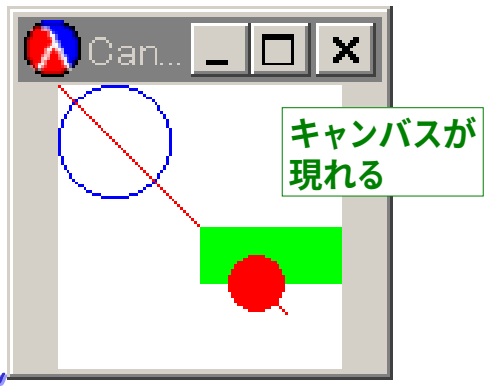
1. 次を「実行用ウインドウ」で実行しなさい

```
(start 100 100)
(draw-solid-line (make-posn 0 0) (make-posn 80 80) 'red)
(draw-solid-rect (make-posn 50 50) 50 20 'green)
(draw-circle (make-posn 20 20) 20 'blue)
(draw-solid-disk (make-posn 70 70) 10 'red)
(stop)
```

実行例



実行例



実習5のまとめ

- キャンバスを開く

```
(start 100 100)
```

- 描画の実行

```
(draw-solid-line (make-posn 0 0) (make-posn 80 80) 'red)
(draw-solid-rect (make-posn 50 50) 50 20 'green)
(draw-circle (make-posn 20 20) 20 'blue)
(draw-solid-disk (make-posn 70 70) 10 'red)
```

posn 構造体のコンストラクタ

- キャンバスを閉じる

```
(stop)
```

Literacy

実習6．ball 構造体を描く

■ ball 構造体を使って，(x, y) の位置に円を描くプログラム **draw-ball** を定義し，実行する

– 構造体とグラフィックス処理の組み合わせ

– ball 構造体のコンストラクタ make-ball とセレクタ ball-x, ball-y を使う

x	位置
y	
delta-x	速度
delta-y	

Literacy

「実習6．ball 構造体を描く」の手順

1. 次を「定義用ウインドウ」で，実行しなさい
   - 入力した後に，Run ボタンを押す

```
(define-struct ball
    (x y delta-x delta-y))
;; draw-ball: ball -> boolean
;; draw a solid disk at (x,y)
(define (draw-ball a-ball)
  (draw-solid-disk (make-posn
             (ball-x a-ball)
             (ball-y a-ball))
           5 'red))
```

2. その後，次を「実行用ウインドウ」で実行しなさい

```
(start 100 100)
(draw-ball (make-ball 10 10 0 0))
(stop)
```

Literacy

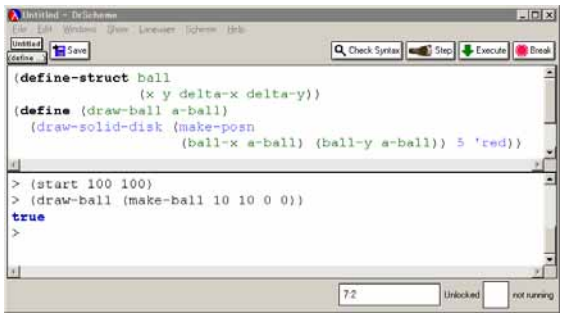
実行例

Literacy

実行例

Literacy

実行例



Literacy

実行例

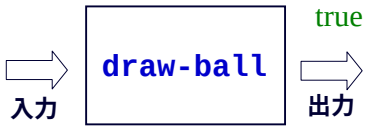
キャンバスに円盤が描かれる

Literacy

draw-ball 関数の入力と出力

a-ball の値:

```
(make-ball 3 4 0 0)
```



入力は ball 構造体

出力は true/false

入力を posn 構造体 (別物) と勘違いしないこと！

Literacy

課題

Literacy

課題1

- 最高気温 high と最低気温 low から，真夏日，夏日，冬日，真冬日を判定する関数 `summer-winter-day` を定義し，実行結果を報告しなさい
  - "Tropical Day"
    (真夏日，1日の最高気温が30度以上の日)
  - "Summer Day"
    (夏日，1日の最高気温が25度以上の日)
  - "Frost Day"
    (冬日，1日の最低気温が0度未満の日)
  - "Ice Day"
    (真冬日，1日の最高気温が0度未満の日)

Literacy

課題2

- ある年 y のある月 m のある日 d が存在するかを調べ，存在すれば d を，存在しなければシンボル '* を返す関数を定義し，実行結果を報告しなさい
  - 例えば，
    2005 10 10 ⇒ 10 を出力
    2005 10 0 ⇒ '* を出力
    2005 10 32 ⇒ '* を出力

Literacy

課題3

- ball 構造体についての問題
  - ball 構造体のデータ `a-ball` の x の値が 0 と 100 の間にあるときに限り true を返し，範囲外 のときには false を返すような関数 `in` を定義し，実行結果を報告しなさい
  - 但し，x = 0 あるいは x = 100 のときには false を返すこと
  - ヒント: 次の空欄を埋めなさい

```
(define-struct ball (x y delta-x delta-y))
(define (in a-ball)
  (cond
    [                                ]
    [else                            ]))
```

Literacy

課題4

- 関数 distance_to_0 についての問題
  - (distance_to_0 (make-ball 4 3 1 1)) から 5 が得られる過程の概略を数行程度で説明しなさい

```
(define-struct ball (x y delta-x delta-y))
;; distance_to_0 : ball -> number
;; to compute the distance of a ball
;; to the origin
;; (distance_to_0 (make-ball 3 4 0 0)) = 5
(define (distance_to_0 a-ball)
   (sqrt
      (+ (sqr (ball-x a-ball))
         (sqr (ball-y a-ball)))))
```

Literacy

課題5

- personal_info 構造体についての問題
  1. 構造体 personal_info 型のデータ a_person の年齢(age)が20以上ならば「'Adult」を，20未満なら「'Child」を出力する関数 check-age2 を定義し，実行結果を報告しなさい
  2. 構造体 personal_info 型のデータ a_person の年齢(age)が20以上ならば true を，20未満なら false を出力する関数 check-age1 を定義し，実行結果を報告しなさい

Literacy