EVERYTHING YOU NEED TO KNOW ABOUT DEFSTRUCT AND DEFMETHOD

Basic usage
1. Defining types
   a. A new type is defined by a call to defstruct
      (defstruct b-tree root left-sub-tree right-sub-tree)
      creates a new type with three fields (root, left-sub-tree, and right-sub-tree)
   b. By defining a new type (such as b-tree), we define the following functions:
      make-b-tree: takes a list of named arguments to set fields
      b-tree-p: takes an object to see if it is a tree
      b-tree-root: takes a tree and returns its root

2. Instantiating types
   a. Once a new type is defined, we can make an instance
      (setq my-tree (make-b-tree :root 1 :left-sub-tree nil))
      We don’t have to give a value for every field; some can be left blank.

3. Setting fields of structures
   a. Now that we have a structure stored in my-tree, we can change its field values:
      (setf (b-tree-root my-tree) 5)

4. Creating generic functions
   a. A generic function is one that executes different code depending on the types of its arguments. We build these by defining a set of methods with the same name that each requires particular data types for its arguments. Defmethod looks like defun except that data types are specified for some or all arguments:
      (defmethod traverse ((x b-tree)) (…))
      (defmethod traverse ((x graph)) (…))
      Would create one generic function (traverse) containing two methods, one for graphs and one for trees.

5. Calling generic functions
   a. We call generic functions just like regular functions. Also, they can be passed as data just like other functions.
      (traverse my-tree)

Advanced capabilities ([http://www.supelec.fr/docs/cltl/clm/node168.html](http://www.supelec.fr/docs/cltl/clm/node168.html) for more details)
Defstruct has a more general form:
( defstruct ( name option-1 option-2 ... option-m)
  doc-string
  slot-description-1
  slot-description-2
  ...
  slot-description-n)
Each slot-description-j is of the form
( slot-name default-init slot-option-name-1 slot-option-value-1 slot-option-name-2 slot-option-value-2 ... slot-option-name-kj slot-option-value-kj)

6. Subtyping
   a. A type can inherit the fields of a supertype with :include
      (defstruct (depth-3-tree (:include b-tree)))

7. Storage
   a. Types are stored as vectors (a primitive data type in Lisp) by default. We can specify that they be stored as lists, then use standard list operations on instances of those types
      (defstruct (b-tree (:type list)) root left right)

8. Constraining field types
   a. Each slot (field) can be given a default value and also constrained to take a particular type
      (defstruct b-tree (root 0 :type integer) (left-sub-tree nil :type tree))
      However, not all implementations of lisp actually produce an error if you violate this type requirement. Alisp (course standard) doesn’t seem to check. However, adding these type requirements give direction to the user of your code.

9. Renaming a structure’s functions
   a. Additional structure functions can be provided using defstruct options. For instance, we can supply an alternative constructor function for b-tree like so:
      (defstruct (b-tree (:constructor create-b-tree (root l r))) root l r)
      Then, we can call (create-b-tree 1 2 3) instead of (make-b-tree :root 1 :l 2 :r 3)